Deeper Learning: Residual Networks, Neural Differential Equations and Transformers, in Theory and Action

Michaël Eli Sander
École normale supérieure, Paris

PhD Thesis
Under the supervision of Gabriel Peyré and Mathieu Blondel

September 25, 2024

This work was supported by the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program (PRAIRIE 3IA Institute) as well as by the European Research Council (ERC project NORIA).

Abstract

This PhD thesis presents contributions to the field of deep learning by exploring the theoretical and practical properties of deep residual neural networks, which have become ubiquitous in state-of-the-art models, from convolutional ResNets to Transformers. Despite their undeniable successes, deep residual networks remain theoretically challenging to understand and are memory-intensive to train. Neural differential equations, the continuous-depth analogs of residual networks, have gained widespread adoption, offering both easier theoretical interpretation and memory-efficient training. However, the formal mathematical correspondence between these discrete and continuous models is still underdeveloped.

In this manuscript, we demonstrate that for a formal correspondence between residual networks and neural ordinary differential equations to hold, the residual functions must be smooth with respect to the network depth. Additionally, we present a result on the implicit regularization of deep residual networks towards neural ordinary differential equations: if the network is initialized as a discretization of a neural ordinary differential equation, then such a discretization holds throughout training. We also consider the use of a discrete adjoint method to train residual neural networks by recovering the activations on the fly through a backward pass of the network, hence avoiding memory costs in the residual layers. We show that this method theoretically and empirically succeeds at large depth.

We then apply this analogy to the design and analysis of novel architectures. First, by incorporating a momentum term, we introduce a drop-in replacement for any residual network that can be trained with equivalent accuracy but with significantly lower memory requirements. These models, termed Momentum Residual Networks, can be interpreted in the infinitesimal step size regime as second-order ordinary differential equations. Second, we demonstrate that Transformers can be interpreted as interacting particle flow maps in the space of probability measures, where the particles represent the tokens. We also investigate the impact of attention map normalization on Transformer behavior, introducing a novel architecture called Sinkformer, in which attention matrices are made doubly stochastic using the Sinkhorn algorithm.

Finally, we provide further contributions to understanding the behavior of Transformers in practice. We examine how causal Transformers perform in-context autoregressive learning on first-order autoregressive processes, decomposing the process into two steps: estimating an inner parameter and predicting the next token. Our theoretical analysis reveals this decomposition for Transformers trained to optimality on such tasks. We also show how to differentiably route tokens to experts in Sparse Mixture of Experts Transformers by introducing new sparse and differentiable top-k operators. This approach leverages a novel formulation of the top-k operator as a linear program over the permutahedron—the convex hull of permutations—and introduces a p-norm regularization term to smooth the operator.

All algorithmic contributions of this thesis are open-sourced and available online.

Résumé

Cette thèse de doctorat apporte des contributions au domaine de l'apprentissage profond en étudiant les propriétés théoriques et pratiques des réseaux de neurones résiduels profonds. Des réseaux ResNets convolutionnels aux Transformers, ces architectures sont omniprésentes dans les modèles d'apprentissage profond de pointe. Malgré leurs succès indéniables, les réseaux résiduels profonds demeurent théoriquement difficiles à analyser et coûteux en mémoire à entrainer. Les désormais populaires équations différentielles neuronales, les analogues en profondeur infinie des réseaux résiduels offrent à la fois une interprétation théorique plus accessible et un coût mémoire significitativement moindre durant l'entraînement. Cependant, comprendre le lien entre les modèles discrets et continus nécessite une fondation mathématique rigoureuse.

Dans ce manuscrit, nous démontrons que, pour qu'une correspondance formelle entre les réseaux résiduels et les équations différentielles neuronales ordinaires soit valide, les fonctions résiduelles doivent être lisses par rapport à la profondeur du réseau. De plus, nous présentons un résultat sur la régularisation implicite des réseaux résiduels profonds vers les équations différentielles neuronales ordinaires : si le réseau est initialisé comme une discrétisation d'une équation différentielle neuronale, alors cette discrétisation se maintient tout au long de l'entraînement. Nous considérons également l'utilisation d'une méthode adjointe discrète pour entraîner les réseaux de neurones résiduels en recalculant les activations à la volée lors d'une rétropropagation dans le réseau, évitant ainsi les coûts mémoires dans les couches résiduelles. Nous montrons que cette méthode réussit théoriquement et empiriquement à grande profondeur.

Ensuite, nous illustrons deux applications de cette analogie en concevant et en étudiant de nouvelles architectures. Tout d'abord, nous proposons, en ajoutant un simple terme d'inertie, une alternative pour tout réseau résiduel qui peut être entraînée avec des performances comparables tout en utilisant significativement moins de mémoire. Ces modèles, appelés Momentum Residual Networks, peuvent être interprétés dans la limite d'un nombre infini de couches comme des équations différentielles ordinaires du second ordre. Ensuite, en interprétant le mécanisme d'attention comme un système de particules en interaction, où les particules représentent les mots (ou tokens). Nous explorons également l'impact de la normalisation des matrices d'attention sur le comportement des Transformers, en introduisant une nouvelle architecture appelée Sinkformer, dans laquelle les matrices d'attention sont rendues doublement stochastiques à l'aide de l'algorithme de Sinkhorn.

Enfin, nous apportons des contributions supplémentaires à la compréhension des Transformers. Nous examinons comment ils effectuent un apprentissage autoregressif à partir d'un contexte sur des processus autoregressifs du premier ordre, décomposant le processus en deux étapes : l'estimation d'un paramètre interne et la prédiction du prochain token. Notre analyse théorique révèle cette décomposition pour les Transformers entraînés de manière optimale sur de telles tâches. Nous montrons également comment acheminer de manière différentiable les tokens vers des experts dans les Transformers de type Sparse Mixture of Experts en introduisant de nouveaux opérateurs top-k parcimonieux et différentiables. Cette approche s'appuie sur une formulation novatrice de l'opérateur top-k comme un programme linéaire sur le permutahedron — l'enveloppe convexe des permutations d'un vecteur — et introduit un terme de régularisation en norme p pour lisser l'opérateur.

Les contributions algorithmiques de cette thèse sont publiquement mis à disposition en ligne.

Contributions and thesis outline

The thesis is structured in three main parts, preceded by an introduction and concluded by a final section. Each part of the manuscript is made up of two chapters, each of which represents an independent contribution and has led to a publication. The chapters are self-contained and the notations may vary from one chapter to another.

Part I: Mathematical foundations for the connection between Residual Neural Networks and Neural Ordinary Differential Equations

This part rigorously studies the formal correspondence between residual networks and neural ordinary differential equations.

[Sander et al., 2022b] *Do Residual Neural Networks Discretize Neural Ordinary Differential Equations?* M.S., Pierre Ablin and Gabriel Peyré. Published at NeurIPS 2022.

[Marion et al., 2024] *Implicit regularization of deep residual networks towards neural ODEs.* Pierre Marion, Yu-Han Wu, M.S., and Gérard Biau. Published at ICLR 2024.

Part II: Analogy between Residual Neural Networks and Neural Ordinary Differential Equations to design and study new architectures

This part uses the analogy between residual networks and neural ordinary differential equations in the wild, that is without strong assumptions on the weights of the networks, to design and study new architectures.

[Sander et al., 2021] *Momentum Residual Neural Networks.* M.S., Pierre Ablin, Mathieu Blondel and Gabriel Peyré. Published at ICML 2021.

[Sander et al., 2022a] *Sinkformers: Transformers with Doubly Stochastic Attention.* M.S., Pierre Ablin, Mathieu Blondel and Gabriel Peyré. Published at AISTATS 2022.

Part III: Transformers in Action

This part presents additional contributions to Transformers, namely how Transformers perform in-context autoregressive learning and how to differentiably route tokens to experts in Sparse Mixture of Experts Transformers.

[Sander et al., 2024] *How do Transformers perform In-Context Autoregressive Learning?.* M.S., Raja Giryes, Taiji Suzuki, Mathieu Blondel, Gabriel Peyré. Published at ICML 2024.

[Sander et al., 2023] *Fast, differentiable and sparse top-k: a convex analysis perspective.* M.S., Joan Puigcerver, Josip Djolonga, Gabriel Peyré, Mathieu Blondel. Published at ICML 2023.

Contents

1 Introduction

This introduction begins by exploring key elements of learning with deep neural networks, focusing on the motivations and frameworks that have fueled their development. Specifically, we focus on residual-based architectures, such as convolutional ResNets and Transformers, and introduce their continuous depth counterpart: neural ODEs. Additionally, we present some challenges faced by modern Transformer models. The following sections provide an overview of the manuscript's contributions, summarizing each chapter's content. Section 1.2 establishes a mathematical foundation connecting residual networks with neural ODEs. Section 1.3 uses this analogy to inspire and evaluate new deep learning models in practical settings. Finally, section 1.4 explores additional insights into Transformers: how they perform next-token prediction in an autoregressive manner and route tokens with experts in sparse mixture of experts models.

Contents

1.1 Learning with deep neural networks

1.1.1 Motivations

It has taken eighty years for artificial neural networks to evolve from theoretical concepts into powerful tools that surpass human performance in various complex tasks. While this may seem like a reasonable amount of time, most of the breakthroughs have actually occurred in the last decade. The exponential growth in computing resources, the vast availability of data, and advances in learning algorithms for training deep neural networks are responsible for the deep learning field's rapid advancement. This revolution has progressed further with large language and diffusion models, demonstrating the immense potential of neural networks to generate coherent and contextually relevant text or high quality images.

The increased capabilities and use of deep learning methods is accompanied by numerous challenges. One major challenge concerns computational resources, which can be both costly and energy-intensive, necessitating the design of memory and compute-efficient models. Another significant issue is the explainability of these models; their complexity often makes them difficult to interpret, leading to transparency issues. Lastly, these models can extrapolate to unseen data with varying degrees of relevance, potentially producing incorrect yet plausible outputs. This calls for a deeper theoretical understanding of the properties of neural networks to design efficient, transparent and accurate models.

Neural networks can be mathematically understood as compositions of differentiable blocks, called layers, each transforming their inputs through a series of simple mathematical operations mainly involving weighted sums (the weights are called parameters) and activation functions. These blocks collectively define a complex, non-linear function of both the network's parameters and its input data. The parameters are tuned on large amounts of data using optimization procedures. The availability of tremendous computational power and the design of highly parallelizable models and pipelines have enabled the scaling of both parameter counts and dataset sizes to unprecedented levels, recently achieving trillions. While this increased complexity may seem to present obstacles to mathematically understand neural networks and address the associated challenges, it actually facilitates the development of frameworks that leverage the vast number of parameters and data to model neural networks in an elegant and useful way.

Importantly, this scaling of parameter counts goes hand in hand with the number of layers increasing to hundreds, defining a large depth limit paradigm. In this context, deep neural networks can be interpreted as ordinary differential equations (ODEs), offering more efficient and transparent models and optimization processes that can be better analyzed.

In the realm of large language models, scaling the dataset size also necessitates the design of neural networks capable of handling an arbitrarily large number of words (or tokens). Among the most notable of these are Transformer models, which can be conveniently interpreted as operators on unordered set of points with varying cardinality, or probability measures, and can be seen as an interacting particle system, where the particles are the tokens.

In addition to providing a playground for modeling deep learning models, both frameworks also serve in the wild as an analogy for developing and understanding new models.

Our ambition in this introduction is to detail these frameworks and present our contributions to them. We will begin with a brief presentation of the (deep) learning framework we are focusing on.

1.1.2 Learning framework

We begin by outlining the fundamental framework for (self-)supervised machine learning.

Dataset. Throughout this manuscript, we explore a learning framework that integrates both supervised and self-supervised learning methodologies. In supervised learning, one is given labeled data to perform a specific task with known outcomes, whereas in self-supervised learning data is unlabeled and the goal is to learn useful representations by solving auxiliary tasks, for instance by predicting part of the input from other parts. Therefore, we consider a dataset of p pairs

$$(x^i, y^i)_{1 \leq i \leq p} \in (\mathcal{X} \times \mathcal{Y})^p,$$

where $\mathcal{X}$ and $\mathcal{Y}$ are subsets of finite dimensional real vector spaces. It is standard to suppose that (x^i, y^i) are realizations of a random variable (X, Y) on $\mathcal{X} \times \mathcal{Y}$.

In this manuscript, we will mostly focus on images, shapes (computer vision) and text (natural language processing (NLP)) for applications of our methods.

Model. We consider a model $F_{\boldsymbol{\theta}}$ parametrized by some parameters $\boldsymbol{\theta}$, also called weights. The objective is to find $\boldsymbol{\theta}^*$ such that:

- In regression (in which case $F_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$) $F_{\boldsymbol{\theta}^*}(x^i) \simeq y^i$ for all $1 \leq i \leq p$.
- In classification (in which case y^i is an integer denoting the class of x^i and $F_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}^{|\mathcal{Y}|}$) we want to use $F_{\boldsymbol{\theta}}$ to form an estimate $p_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}^{|\mathcal{Y}|}$ of the posterior probability distribution $p(Y|X)$ over the classes, and find $\boldsymbol{\theta}^*$ such that $p_{\boldsymbol{\theta}^*}(x^i)_{y^i}$ is maximal for all $1 \leq i \leq p$.

Loss. To quantify such approximation, we consider that we have a loss function ℓ which is small when $F_{\boldsymbol{\theta}}(x^i) \simeq y^i$ or when $p_{\boldsymbol{\theta}}(x^i)_{y^i}$ is maximal. In regression, one can consider the empirical mean squared error

$$\ell(\boldsymbol{\theta}) = \frac{1}{p} \sum_{i=1}^{p} \|F_{\boldsymbol{\theta}}(x^i) - y^i\|^2.$$

Similarly, the mean squared error can be considered in denoising diffusion models (Hyvärinen and Dayan, 2005; Song and Ermon, 2019; Song et al., 2020), where the loss quantifies the accuracy of reconstructing an original image from a noisy version.

In classification, one can consider the cross-entropy loss:

$$\ell(\boldsymbol{\theta}) = -\frac{1}{p} \sum_{i=1}^{p} \log[p_{\boldsymbol{\theta}}(x^i)]_{y^i} \quad \text{with} \quad p_{\boldsymbol{\theta}}(x^i) = \frac{\exp(F_{\boldsymbol{\theta}}(x))}{\sum_{k=1}^{|\mathcal{Y}|} \exp([F_{\boldsymbol{\theta}}(x)]_k)},$$

This loss is also used to measure how well a model predicts the next token in a sequence given the preceding tokens in a next-token prediction framework.

Learning. The goal of the learning procedure is therefore to find $\boldsymbol{\theta}^*$ that minimizes ℓ, that is $\boldsymbol{\theta}^* \in \operatorname{argmin}(\ell)$. Without further assumptions than differentiability on the model $F_{\boldsymbol{\theta}}$, the common approach is to rely on gradient-based optimization methods to iteratively approach $\boldsymbol{\theta}^*$. The most common approach is gradient descent or stochastic gradient descent, though more sophisticated approaches are also commonly used in practice such as Adam (Kingma and Ba, 2014). The iterations for gradient descent are, starting with some initialization $\boldsymbol{\theta}(0)$:

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta(t) \nabla \ell(\boldsymbol{\theta}(t)), \tag{1.1}$$

where $\eta(t)$ is the step-size or learning rate. In this manuscript, we will also consider the infinitesimal step-size regime of equation (1.1), known as gradient flow:

$$\frac{d\boldsymbol{\theta}(t)}{dt} = -\nabla \ell(\boldsymbol{\theta}(t)). \tag{1.2}$$

The question of global convergence of these methods as well as the impact of the choice of the initialization on such convergence is an active area of research. We discuss such properties in sections 1.2.1, 1.2.2 and 1.4.1.

Deep Neural Networks. Throughout this manuscript, we are going to consider $F_{\boldsymbol{\theta}}$ to be a deep neural network. Without loss of generality, $F_{\boldsymbol{\theta}}$ can be modeled as, starting from a data point $x_0 \coloneqq x$:

$$x_{n+1} = F_n(x_n, \theta_n), \quad 0 \leq n \leq N-1, \tag{1.3}$$

where N is the number of layers or depth of the network, $\boldsymbol{\theta} = (\theta_0, \dots, \theta_{N-1})$ and the F_n's are parametrized functions. One then has $F_{\boldsymbol{\theta}}(x) = x_N$. The x_n's are called activations. Many designs are possible for F_n, and this manuscript will consider three of them, namely perceptrons, convolutional and attention layers.

While these parameterized functions form the backbone of modern deep neural networks, additional components are essential for achieving optimal performance as the depth N increases. The first, which is not the focus of this manuscript, is the use of normalization layers at each iteration to ensure effective signal propagation and avoid vanishing/exploding gradients. Normalization layers enable convergence of neural networks with dozens of layers. However, when one considers even deeper networks, a second ingredient, which is central to this thesis, is the use of skip connections, ensuring the principle that *the deeper, the better*. A deep neural network using skip connections is called *residual neural network.*

1.1.3 Residual Neural Networks

It has been empirically observed (Srivastava et al., 2015; He and Sun, 2015) that the error of deep convolutional networks in image classification tasks first saturates with depth and then rapidly degrades. This phenomenon is paradoxical because if the F_n in (1.3) are expressive enough to express the identity mapping, one can easily construct the deeper counterpart of an existing model by stacking identity layers on top of it. In this way, a deeper model is at least as expressive as a shallower one. However, as the parameters $\boldsymbol{\theta}$ are learned by gradient based methods, such desirable solutions do not seem to be achieved in practice. The idea of He et al. (2016a) is to solve this paradox by explicitly letting some of the F_n's depend on a residual mapping f_n: $F_n(x, \theta_n) = x + f_n(x, \theta_n)$ for some values of n. Note that this is possible if and only if $f_n(x, \theta_n)$ has the same shape as x. With such a formulation, for F_n to be close to the identity, it suffices for f_n to be close to zero, which happens to be easier from an optimization perspective. In practice, this small change is enough to ensure that stacking more layers improves accuracy, and the original idea of residual learning in convolutional neural networks (He et al., 2016a,b) enabled the resulting architecture, called ResNet, to beat all existing deep learning models on the challenging ImageNet (Deng et al., 2009a) classification task. It is important to note that a standard ResNet consists of two core components. The first are downsampling layers, which halve the size of the current activation x_n and double the number of filters. There are usually 4 downsampling layers in standard ResNets. In between two of such layers with indices n_k and n_{k+1} are the residual layers themselves, where each neural network f_n in (1.3) is structurally identical: $f_n \coloneqq f$ and one iterates:

$$x_{n+1} = x_n + f(x_n, \theta_n), \quad n_k \leq n \leq n_{k+1} - 1.$$

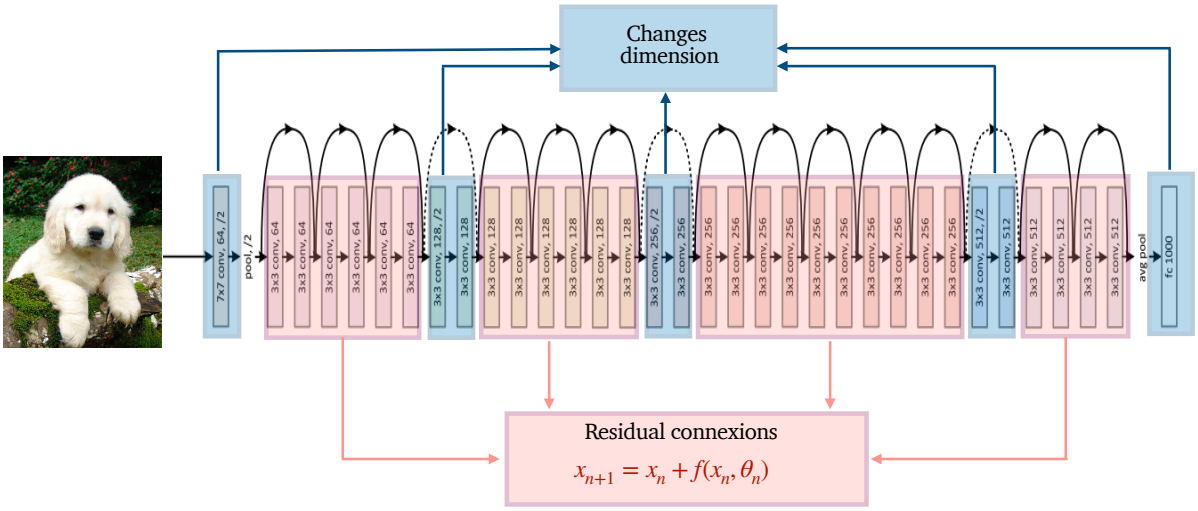

Figure 1.1: **ResNet 34 architecture** as presented in He et al. (2016a). We highlight in blue the downsampling layers for which the dimension changes, and in red the residual layers.

The presence of these two components is illustrated in Figure 1.1. This remark being made, we will, in the rest of this introduction solely focus on the residual layers and consider that our deep network $F_{\boldsymbol{\theta}}$ simply iterates

$$x_{n+1} = x_n + f(x_n, \theta_n), \quad 0 \leq n \leq N-1. \tag{1.4}$$

Since their introduction in the context of convolutions, residual connections are now ubiquitous in the deep learning community in various forms.

In this manuscript, we will consider the following choices for f, which we describe in their simplest form as follows:

- **Perceptron layers** (Rosenblatt et al., 1962; Amari, 1967): in this case $f(x, \theta) = V\sigma(Wx)$, where $x \in \mathbb{R}^d$, W and V are real matrices and $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linearity applied component-wise. We consider such models in Chapters 3 and 7.
- **Convolutional layers** (LeCun et al., 1998): in this case $f(x, \theta) = V * \sigma(W * x)$, where $x \in \mathbb{R}^{d_1 \times d_2 \times C}$, C is the number of channels in x, W and V are convolutional filters, $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linearity applied component-wise and $*$ is the multi-channel convolution operator, defined as $(W * x)(i, j, k) = \sum_{m=1}^{d_1} \sum_{n=1}^{d_2} \sum_{c=1}^{C} W(m, n, f, k) \cdot x(i + m, j + n, c)$ with $W \in \mathbb{R}^{d_1 \times d_2 \times C \times K}$. We consider such models in Chapters 2, 3 and 4.
- **Attention layers** (Bahdanau et al., 2014a; Vaswani et al., 2017): in this case, $f(x, \theta) = \sigma(xW_Q W_K^\top x^\top)xW_V$, where $x \in \mathbb{R}^{T \times d}$ is a sequence of T tokens, W_Q, W_K, and W_V are real matrices and σ is the row-wise SoftMax operator. More details on such layers, which are at the core of Transformer models, are provided in Section 1.1.6. We consider such models in Chapters 5, 6 and 7.

Throughout this manuscript, we will use the terminologies "residual neural network", "residual network" and "ResNet" to refer to a model of the form (1.4).

1.1.4 Challenges with Deep Residual Neural Networks

We saw how residual connections enable stacking many layers. In practice, depth reaches thousands in computer vision (He et al., 2016b) and hundreds in NLP (Brown et al., 2020).

However there are two main issues when considering deep models, namely:

- Theoretical understandings of deep neural networks are difficult.
- Deep neural networks are memory costly to train.

Theoretical hardness. Most of the theoretical analysis regarding the approximation capacities, statistical properties and optimization behavior of neural networks are dedicated to the one-layer (shallow) case. While universal approximations results and approximation rates can still be derived for deep networks (DeVore et al., 2021), generalization bounds for deep neural networks increase polynomially with depth (Bartlett et al., 2017), and most results on the analysis of convergence of gradient descents to global minima in the rich regime (that is when all the parameters significantly evolve during the optimization process) concern the shallow case (Chizat and Bach, 2020).

Memory cost. The memory cost of training deep architectures arises from the *backpropagation* algorithm, which is the preferred method for computing the gradient of the scalar-valued function $\ell(\boldsymbol{\theta})$. This algorithm employs the chain rule with a backward traversal of the computational graph (Bauer, 1974) and is also known as reverse-mode automatic differentiation (Baydin et al., 2018). The computational cost is comparable to that of evaluating the function itself. To backpropagate gradients through a neural architecture without additional assumptions, all intermediate activations must be stored during the forward pass. This approach is implemented in popular deep learning libraries such as Pytorch (Paszke et al., 2017), Tensorflow (Abadi et al., 2016), and JAX (Jacobsen et al., 2018). The backpropagation equations, for which the concept is illustrated in Figure 1.2 are

$$\nabla_{\theta_{n-1}}\ell = [\partial_\theta f(x_{n-1},\theta_{n-1})]^\top \nabla_{x_n}\ell, \quad \nabla_{x_{n-1}}\ell = [I + \partial_x f(x_{n-1},\theta_{n-1})]^\top \nabla_{x_n}\ell. \tag{1.5}$$

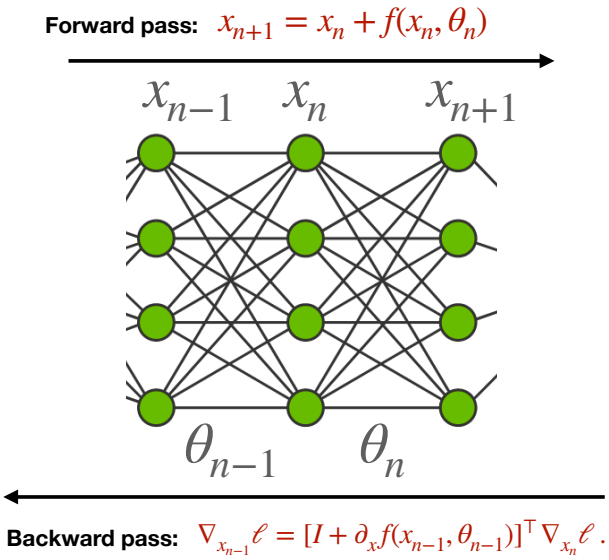

Figure 1.2: **Illustration** of the backpropagation algorithm.

The computations in (1.5) necessitate the storage of the activations x_n's. This can cause memory issues in increasingly deep architectures because the memory requirements to store the x_n's is simply too big, as illustrated in Figure 1.3.

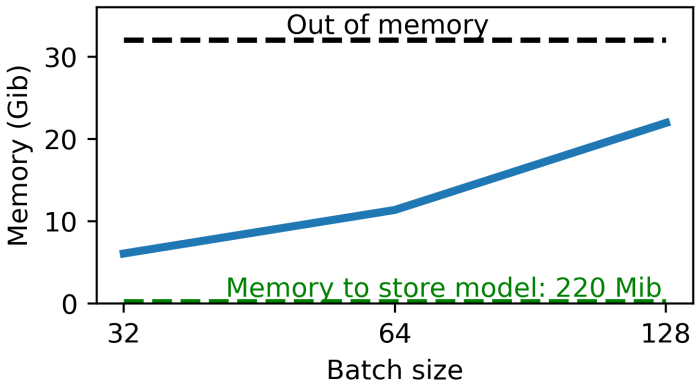

Figure 1.3: **Memory needed** (in blue) to compute gradients as the function of the number of samples processed at the same time (batch size) for a convolutional ResNet 152 on ImageNet (Deng et al., 2009a). In contrast, the memory needed to store the parameters of the model is negligible..

1.1.5 Addressing Challenges with Neural ODEs

We now present an infinite-depth limit viewpoint on residual neural networks, called neural ODEs, in order to tackle these challenges.

The idea behind Neural Ordinary Differential Equations (Weinan, 2017a; Chen et al., 2018) is to interpret (1.4) as a Euler discretization of the ODE

$$\frac{\mathrm{d}X}{\mathrm{d}s} = f(X(s), \Theta(s)) \quad \text{with} \quad X(0) = x_0. \tag{1.6}$$

While a residual neural network consists in a succession of discrete transformation of the activation x_n and outputs x_N, the Neural ODE corresponds to a smooth transformation mapping an initial condition $X(0)$ to its deep representation $X(1)$, that is the solution of (1.6) at some fixed time horizon, that we take equal to 1 without loss of generality. This comparison is illustrated in Figure 1.4. Note that for the choices of f made in this manuscript, an analytical solution to the neural ODE (1.6) will never be available and one has to rely on numerical solvers to solve the neural ODE (Kidger, 2022), for which dedicated software is available (Chen, 2018). It has been shown that the performance of a given neural ODE depends on the numerical integrator for solving it (Gusak et al., 2020). See Kidger (2022) for a complete survey on neural ODEs.

Neural ODEs enjoy theoretical and practical advantages compared to finite depth residual neural networks.

Theoretical advantages. First, the approximation capabilities of neural ODEs are well understood (Teshima et al., 2020; Zhang et al., 2020a): a neural ODE is a universal approximator of a large class of diffeomorphisms, that is continuous invertible maps with continuous inverse. Secondly, while standard generalization bounds of neural networks grow with depth (Bartlett et al., 2019), it is still possible to derive generalization bounds for neural ODEs (Hanson and Raginsky, 2022; Marion, 2023). Last, global convergence results of gradient based optimization methods can be derived for neural ODEs (Barboni et al., 2022, 2024).

Practical advantages: memory-free training. Practically, neural ODEs offer the significant advantages of a memory-efficient training (Chen et al., 2018). More precisely, the continuous

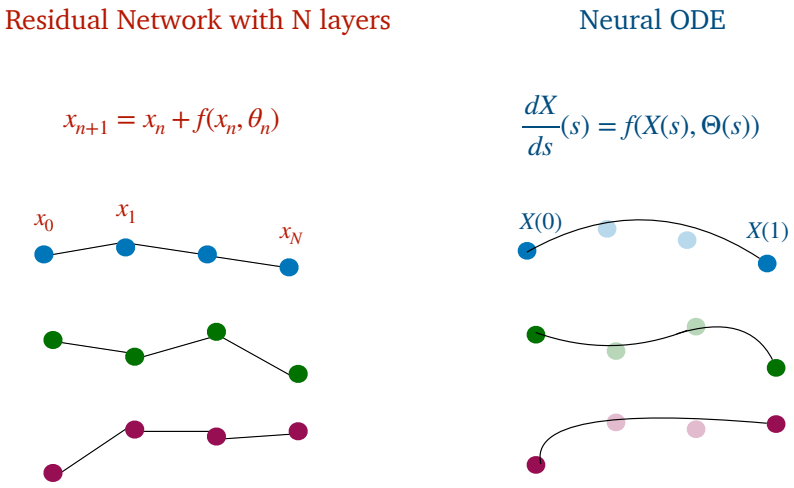

Figure 1.4: **Comparison** between residual neural networks and Neural ODEs. While the first map an input to a deep representation through a finite series of transformations, the second define this deep representation implicitly as the solution of the ODE at time 1.

adjoint state method (Pontryagin, 1987; Chen et al., 2018) gives

$$\nabla_{\Theta(s)}\ell = \partial_\Theta[f(X(s),\Theta(s))]^\top \nabla_{X(s)}\ell \mathrm{d}s, \quad \frac{\mathrm{d}\nabla_{X(s)}\ell}{\mathrm{d}s} = -[\partial_X f(X(s),\Theta(s))]^\top \nabla_{X(s)}\ell. \tag{1.7}$$

The key advantage of using equation (1.7) is that one can recover $X(s)$ on the fly by solving the Neural ODE (1.6) backward in time starting from $X(1)$. This strategy avoids storing the forward trajectory $(X(s))_{s\in[0,1]}$ and leads to a $O(1)$ memory footprint (Chen et al., 2018). However, numerical discretization errors when solving the ODE backward will lead to a difference between the forward and backward trajectory, such that the gradients obtained in equation (1.7) will not be identical to the ones obtained by differentiating through the numerical solver. These stability issues can lead to failures of the continuous adjoint method (worse accuracy compared to differentiating through the solver, significantly slower training) (Gholami et al., 2019; Teh et al., 2019; Kidger, 2022).

Applications. Neural ODEs have several key applications, particularly in dynamic modeling and time-series prediction. They are widely used in physics-informed modeling, continuous-time modeling of dynamical systems (Greydanus et al., 2019), and generative modeling (Chen et al., 2018). Additionally, they are applied in system identification, population dynamics, financial modeling. However, while the parametrization of f using a continuous depth dependent parameter $\Theta(s)$ may offer parameter efficiency, it greatly decreases the performance of neural ODEs on standard supervised-learning tasks such as ImageNet classification (Zhuang et al., 2021).

Correspondence between residual networks and neural ODEs. Even though neural ODEs are presented as the infinite depth counterpart of residual networks, the fact that they are used in different applications calls for a better understanding of the true correspondence holding between them. Under which conditions trained residual networks behave like neural ODEs? Answering this question would offer a more rigorous foundation for the widely held belief that residual networks and neural ODEs behave similarly in the large-depth limit (see, e.g., Haber and Ruthotto, 2017a; E et al., 2019; Dong et al., 2020; Massaroli et al., 2020; Kidger, 2022). Secondly, if trained residual networks can indeed be viewed as discretizations of neural ODEs, then one can use the extensive theoretical and practical arsenal of ODE theory to residual networks. Last,

but not least, residual networks would benefit the practical advantage of neural ODEs, that is their reduced memory training. We further discuss this question in Section 1.2, and in Chapters 2 and 3 of this manuscript.

However, scaling deep learning methods requires understanding not only their behavior in the large depth limit but also in the large dataset limit. This is particularly crucial when considering Transformers, which are designed to handle an arbitrarily large number of tokens. We will now introduce this architecture.

1.1.6 Transformers

Transformers (Vaswani et al., 2017) have emerged as the state-of-the-art architecture in natural language processing (NLP) tasks (Devlin et al., 2018) and have also demonstrated remarkable performance in computer vision (Dosovitskiy et al., 2020). These models now serve as the foundation for large-scale language models, such as GPT (Radford et al., 2018; Brown et al., 2020), which have significantly advanced the capabilities of artificial intelligence in both understanding and generating human language.

Typical pipeline. The fundamental operation of Transformers involves processing sequences of tokens $(s_1, \dots, s_T)$ of arbitrary length T. Initially, the tokens are embedded to create a sequence $X = (X_1, \dots, X_T)$, which represents the input data in a real-valued vector space. To incorporate positional information, a sequence-independent positional encoding is either concatenated with or added to each token embedding, resulting in a new sequence $x = (\mathrm{x}_1, \dots, \mathrm{x}_T)$. This sequence is then processed through a series of Transformer blocks, each equipped with residual connections (He et al., 2016a), which we recall help to preserve the flow of gradients and stabilize training.

Each Transformer block consists of three primary components:

- A **multi-head self-attention mechanism** layer with residual connection: The update is performed as $x \leftarrow x + f(x, \theta)$, where f is the multi-head self-attention mechanism described in equation (1.8).
- A **feedforward multi-layer perceptron** (MLP) with residual connection: This typically consists of a 2-layer perceptron with a ReLU non-linearity, updating each token as: $\mathrm{x}_t \leftarrow \mathrm{x}_t + \sigma(W_2(\sigma(W_1 \mathrm{x}_t)))$.
- Two **layer normalization** layers: One layer norm is applied after the multi-head self-attention and another after the feedforward MLP, updating each token as: $\mathrm{x}_t \leftarrow \frac{\mathrm{x}_t - \mu_t}{\sigma_t} \cdot \gamma + \beta$, where μ_t and σ_t are the mean and standard deviation of x_t across its features, and γ, β are learnable.

While the feedforward layer and normalization layers operate on each token independently, the multi-head self-attention mechanism allows the model to mix information across different tokens, thereby capturing dependencies and contextual relationships within the sequence. This mechanism involves applying multiple self-attention operations in parallel and is parametrized by a set of weight matrices $(W_Q^h, W_K^h, W_V^h, W_O^h)_{1 \le h \le H}$, where H denotes the number of attention heads (Vaswani et al., 2017; Michel et al., 2019). The output of the multi-head self-attention mechanism is given by:

$$f(x, \theta) = \left(\sum_{h=1}^{H} W_O^h \sum_{t'=1}^{T} \mathcal{A}_{t,t'}^h W_V^h \mathrm{x}_{t'} \right)_{t \in \{1, \dots, T\}}, \tag{1.8}$$

where $\mathcal{A}^h$, the attention matrix, determines the attention weights between tokens and is typically defined as:

$$\mathcal{A}^h_{t,t'} = \frac{e^{\langle W^h_Q \mathrm{x}_t, W^h_K \mathrm{x}_{t'} \rangle} m_{t,t'}}{\sum_{\tau=1}^T e^{\langle W^h_Q \mathrm{x}_t, W^h_K \mathrm{x}_\tau \rangle} m_{t,\tau}} = \mathrm{SoftMax}(\langle W^h_Q \mathrm{x}_t, W^h_K \mathrm{x}_\cdot \rangle m_{t,:}).$$

In this formulation, $m_{t,t'}$ can act as a masking function, and can either be set to 1 (indicating that all tokens are attended to equally, as in an *encoder* model) or $1_{t \geq t'}$ (imposing a causal structure where only past and present tokens are attended to, as in a *decoder* model). This flexibility in attention mechanisms allows Transformers to be tailored for different tasks, whether requiring bidirectional context understanding or autoregressive sequence generation.

Following this succession of residual-based attention and feedforward blocks there is a classification or regression head which typically takes the form of a feedforward neural network and usually acts on a pooled transformation of the output sequence or on one particular token in the output sequence. More specifically, in tasks such as text classification, a common approach is to apply a pooling operation to the output sequence to summarize the information into a single vector. One common technique is "mean pooling", where the mean of all tokens' representations in the output sequence is taken. Alternatively, max pooling can be used. Another common approach is to use a specific token's output, such as the [CLS] (classification) token. This token is often designed to capture global information about the entire sequence, and its final hidden state is used as the input to the classification or regression head. This is typically done in Vision Transformers (Dosovitskiy et al., 2020). It is also the case in next-token prediction, where the last token iteratively plays the role of classification token, and the model $F_{\boldsymbol{\theta}}$ is used to estimate a conditional probability $p(S_{t+1} = s_{t+1} | S_1 = s_1, \cdots, S_t = s_t)$. In this case, the tokens correspond to indexes in a finite vocabulary of size V and the loss $\ell(\boldsymbol{\theta})$ takes the form of a cross entropy loss:

$$\ell(\boldsymbol{\theta}) = -\frac{1}{p} \sum_{i=1}^p \left(\frac{1}{T_i} \sum_{t=1}^{T_i} \log p_{\boldsymbol{\theta}}(s^i_1, \cdots, s^i_t)_{s^i_{t+1}} \right) \quad \text{with} \quad p_{\boldsymbol{\theta}}(s^i_1, \cdots, s^i_t) = \frac{\exp\left(F_{\boldsymbol{\theta}}(x^i)_t\right)}{\sum_{s=1}^V \exp\left(F_{\boldsymbol{\theta}}(x^i)_t\right)_s}.$$

An illustration of the Transformer's typical pipeline is provided in Figures 1.5 and 1.6.

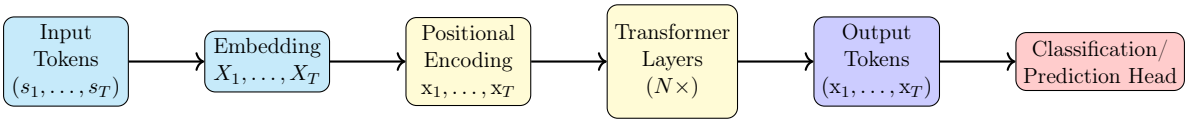

Figure 1.5: Global architecture of the Transformer model of depth N.

Successes and variants. Transformers have revolutionized the deep learning field with successes in text generation (Team et al., 2023; Jiang et al., 2023; Touvron et al., 2023), computer vision (Dosovitskiy et al., 2020; Tu et al., 2022), protein structure prediction (Jumper et al., 2021), audio generation (Borsos et al., 2023), and music generation (Agostinelli et al., 2023). Transformers also form the backbone of self-supervised learning models, enabling representation learning across different modalities (see Balestriero et al., 2023 for a detailed survey).

The vast range of Transformer applications has fueled the development of numerous variants aimed at enhancing Transformer performance, particularly by refining the attention mechanism. For instance, linear attention (Katharopoulos et al., 2020) replaces the SoftMax function in (1.8) with the identity function, enabling faster inference (Katharopoulos et al., 2020; Fournier

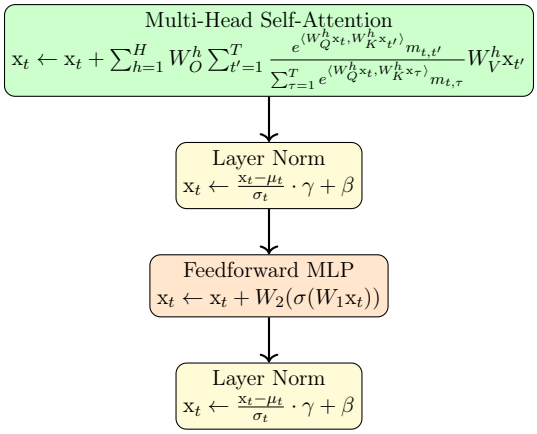

Figure 1.6: Detailed view of a single Transformer layer with Multi-Head Attention, Feedforward MLP, and Layer Norms. Formulas for Multi-Head Attention, LayerNorm, and MLP are included.

et al., 2023). Additionally, Kim et al. (2021) replace the dot-product $\langle W_Q^h \mathrm{x}_t, W_K^h \mathrm{x}_{t'} \rangle$ in (1.8) with an L_2 norm cost, $\|W_Q^h \mathrm{x}_t - W_K^h \mathrm{x}_{t'}\|^2$, to create Lipschitz continuous and invertible attention modules. Furthermore, Wortsman et al. (2023) replace the SoftMax function in (1.8) in vision Transformers with a ReLU activation, experimentally demonstrating that this adjustment can match the performance of standard SoftMax-based Transformers.

One contribution of this manuscript is the proposal to replace the row-wise normalization of the attention matrix due to the SoftMax with a doubly stochastic normalization, using Sinkhorn algorithm (Sinkhorn, 1964; Cuturi, 2013). This modification is motivated by the empirical observation that attention matrices tend to approximate doubly stochastic matrices during training. The resulting architecture, Sinkformer (Sander et al., 2022a), is presented in Section 1.3.2 and Chapter 5. We theoretically analyze this architecture in the regime of an arbitrary high number of tokens, described in the following paragraph.

Arbitrary number of tokens. One striking aspect of Transformers is that formulation (1.8), which is the only operation involving interactions between different tokens, is valid for an arbitrary number of tokens. Typically, T can therefore theoretically be as large as desired, and in practice, T can reach 2 million in most recent large language models, such as Gemini (Team et al., 2023).

In addition, in the unmasked case (that is when $m_{t,t'} = 1$), it is easily seen that the map $f(., \theta)$ in equation (1.8) is permutation equivariant. More formally, for any permutation σ, defining $x_\sigma \coloneqq (\mathrm{x}_{\sigma(1)}, \cdots, \mathrm{x}_{\sigma(T)}))$, we have $f(\mathrm{x}_\sigma, \theta) = f(\mathrm{x}, \theta)_\sigma$. An appealing approach to modeling such equivariant architectures is to consider them as operating on probability measures or point clouds of varying cardinality (De Bie et al., 2019; Vuckovic et al., 2021; Zweig and Bruna, 2021). Specifically, a collection of points $(\mathrm{x}_t)_{1 \leq t \leq T}$, where $\mathrm{x}_t \in \mathbb{R}^d$, can be interpreted as a discrete measure on $\mathbb{R}^d$:

$$\mu \coloneqq \frac{1}{T} \sum_{t=1}^{T} \delta_{\mathrm{x}_t} \in \mathcal{M}(\mathbb{R}^d),$$

where $\mathcal{M}(\mathbb{R}^d)$ denotes the set of probability measures on $\mathbb{R}^d$. In this context, a map Γ_μ acts on μ by transforming it into another measure:

$$\mu \mapsto \frac{1}{T} \sum_{t=1}^{T} \delta_{\Gamma_\mu(\mathrm{x}_t)}.$$

This perspective is particularly useful for handling unordered sets of points, allowing for the analysis of their evolution under transformations. Moreover, it facilitates the exploration of the mean field (or large sample) limit, where $T \to \infty$, which is essential for conducting theoretical analysis. This approach is analogous to analyzing gradient descent in the mean-field limit, offering insights into the behavior of systems as the number of its parameters becomes large (Chizat and Bach, 2020).

One main contribution of this manuscript, discussed further in section 1.3.2 and developed in Chapter 5 is to adapt this framework to Transformers, and also to define the neural ODE corresponding to stacking self-attention layers. We will see that such a neural ODE takes the form of a partial differential equation (PDE)

$$\partial_s \mu + \operatorname{div}(\mu \Gamma_\mu) = 0.$$

Viewing transformers as particles in interaction systems was done in Lu et al. (2019). This formulation of transformers as maps between probability measures as well as the PDE interpretation was introduced in Sander et al. (2022a) (Chapter 5 of this manuscript). Then, it has been used to derive clustering results (Geshkovski et al., 2023), smoothness guarantees (Castin et al., 2024) as well as universality results (Furuya et al., 2024). The formalism of Transformers as maps between probability measures was also extended to the masked case by Castin et al. (2024) by considering augmented tokens $(\mathrm{x}_t, t) \in \mathbb{R}^{d+1}$.

1.1.7 Transformers in action

1.1.7.1 Autoregressive In-Context Learning.

The primary domain where Transformers have seen widespread success recently is in next-token prediction within large language models. Despite their remarkable achievements in autoregressive tasks, the underlying reasons for the effectiveness of Transformers in this setting still lack a complete theoretical understanding. One important question concerns the expressivity of Transformers for next-token prediction.

Universality of encoder-only Transformers. The universal approximation properties of encoder-only Transformers are well established. Yun et al. (2019); Nath et al. (2024); Furuya et al. (2024) demonstrate that Transformers can universally approximate permutation-equivariant functions. A more constructive approach, though applicable to a narrower class of functions, is proposed by Wang and E (2024), offering deeper insights into the mechanisms governing Transformer expressivity.

In-context learning. When it comes to decoder-only models, another popular recent line of works consists in theoretically understanding the in-context learning ability of Transformers. Indeed, the seminal work by Brown et al. (2020) introduced the in-context learning phenomenon observed in Transformer language models: these models are capable of solving few-shot learning tasks using examples provided within the context. Specifically, given a sequence $(\mathrm{x}_1, \varphi(\mathrm{x}_1), \mathrm{x}_2, \varphi(\mathrm{x}_2), \cdots, \mathrm{x}_k)$, a trained Transformer can infer the next output $\varphi(\mathrm{x}_k)$ without requiring further parameter updates. This unexpected capability has spurred a wave of recent research. Some studies have explored the ability of SoftMax attention Transformers to perform in-context learning, without theoretically studying the training dynamics (Garg et al., 2022; Akyürek et al., 2022; Li et al., 2023). Others have focused on linear attention (where the SoftMax is replaced by the identity in equation (1.8)) and analyzed the minimizers of the training loss when φ is sampled from linear forms in $\mathbb{R}^d$, specifically where $\varphi(\mathrm{x}) = w^\top \mathrm{x}$ for some w (Mahankali

et al., 2023; Ahn et al., 2023; Zhang et al., 2023). These works, in particular, examine the potential of Transformers to implement optimization algorithms during their forward pass at inference in order to estimate w, as suggested empirically by Von Oswald et al. (2023a). A contribution of this manuscript (Chapter 6) is to start bridging the gap between the above formulation and analysis of in-context learning with what we will refer to as autoregressive in-context learning.

Autoregressive in-context learning. Recent works (Von Oswald et al., 2023b; Sander et al., 2024; Nichani et al., 2024) have begun to provide insights into the expressivity of Transformers for general next-token prediction tasks. In Von Oswald et al. (2023b), the authors explore the hypothesis that, similar to in-context learning, a trained Transformer predicts the next token by first estimating an internal parameter and then using it to output a prediction for s_{t+1}. A simplified model (Sander et al., 2024), presented in Chapter 6, which generalizes the work of Von Oswald et al. (2023b), assumes that tokens satisfy the relation $s_{t+1} = \varphi_W(s_{1:t})$, where W is a context-dependent parameter that varies with each sequence. We explore the hypothesis that for a model $\mathcal{T}_{\boldsymbol{\theta}}$ to perform well in mapping $s_{1:t}$ to s_{t+1}, it first estimates W and then predicts the next token. Intuitively, because W varies with each sequence, the accuracy of predicting W heavily depends on the attention mechanism's ability to adapt its computations based on the context $s_{1:t}$.

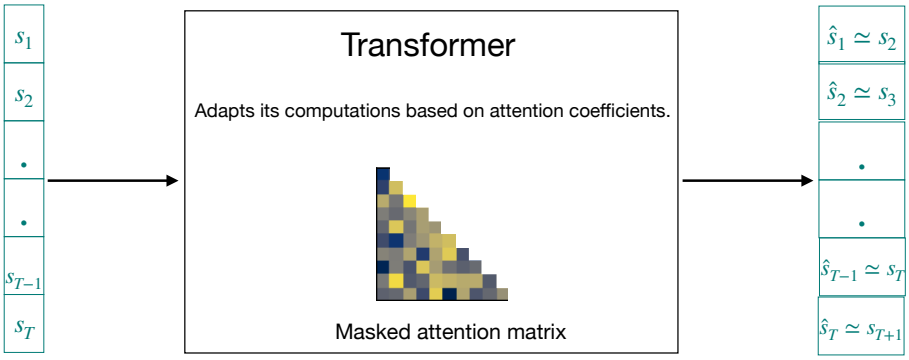

Figure 1.7: **Illustration** of the autoregressive setting. Given any subsequence $s_{1:t}$, the model is trained to output a prediction of the next token s_{t+1}.

1.1.7.2 Sparse Mixture of Experts with Top-k Gating.

As the number of parameters in Transformer models increases, their performance tends to improve, but so does the associated computational cost. This trade-off is common in deep learning; however, the structure of Transformers offers opportunities to mitigate these costs. Notably, a significant portion of a Transformer's parameters—up to 90% in some models like PaLM (Chowdhery et al., 2023)—reside in the feedforward networks. Importantly, these feedforward layers are applied independently to each token, making it possible to consider different feedforward networks for different tokens. The idea behind sparse mixture of experts (MoEs) (Shazeer et al., 2017; Riquelme et al., 2021) is to replace some or each feedforward layer with a set of smaller specialized layers, called experts. Typically, only a subset of these feedforward networks is activated for each token, based on a gating mechanism's selection. This approach allows the model to maintain or even enhance its performance while significantly reducing the number of active parameters during training and inference. Sparse MoEs are successfully used in computer vision (Dehghani et al., 2023) and language modeling (Jiang et al., 2024). For a detailed review on sparse mixture

of experts in the context of computer vision, see Liu et al. (2024).

As explained in Liu et al. (2024), the most common formulation consists of gating each token to k experts through a parametrized function $\mathrm{Gate}(\mathrm{x}) := \text{top-k}\left(\mathrm{SoftMax}\left(W\mathrm{x}\right)\right)$. The operator top-k is defined as top-k(y), which keeps all the k largest coordinates of y intact and sets to 0 all the others. A MoE layer with E experts $(\mathrm{MLP}_r)_{1\leq r\leq E}$ is then defined as

$$\mathrm{MoE}(\mathrm{x}) := \sum_{r=1}^{E} \mathrm{Gate}(\mathrm{x})_r \cdot \mathrm{MLP}_r(\mathrm{x}),$$

and can be used in place of any standard MLP layer in large scale Transformers. This process is further illustrated in Figure 1.8.

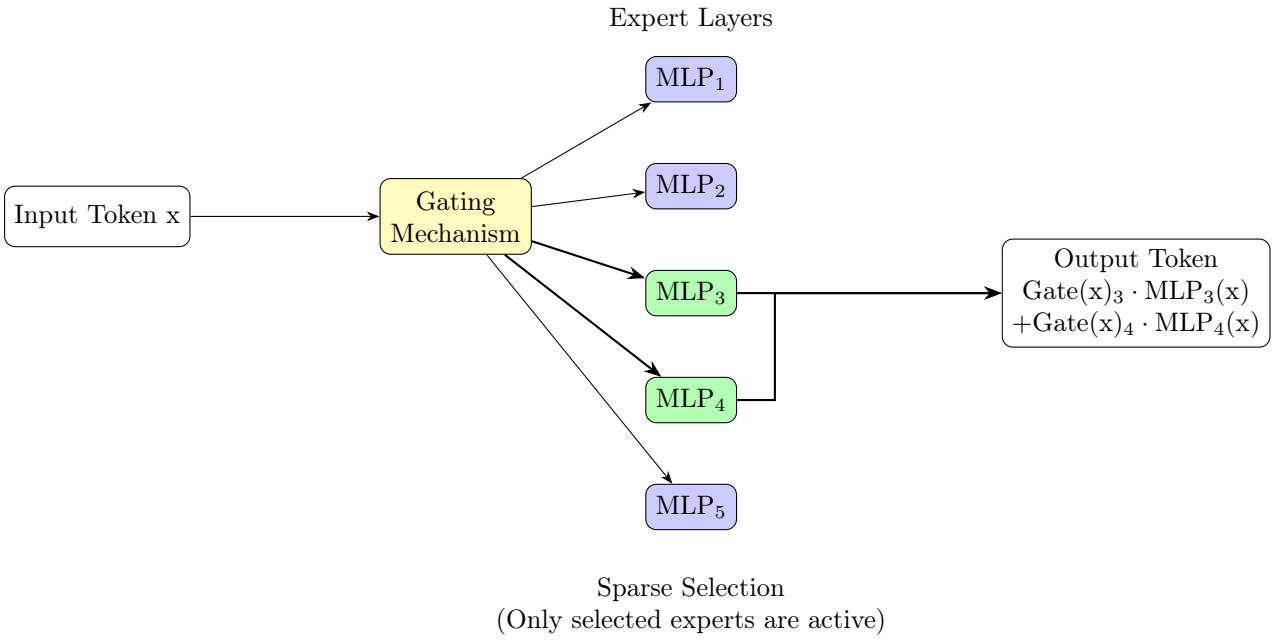

Figure 1.8: Illustration of Sparse Mixture of Experts in Transformers. The input token is processed by a subset of expert layers, selected through a gating mechanism, to generate the output token.

An important drawback of this gating procedure is that the top-k operator is a discontinuous piecewise affine function with derivatives that are either undefined or constant. Since deep learning models are typically trained end-to-end with gradient backpropagation, an interesting problem is to propose a differentiable relaxation of the top-k operator, extending recent advances in proposing differentiable relaxations of standard operators and algorithms (Cuturi et al., 2019; Blondel et al., 2020b; Berthet et al., 2020; Grover et al., 2019; Prillo and Eisenschlos, 2020; Blondel and Roulet, 2024). Existing approaches already consider differentiable top-k operators (Amos et al., 2019; Qian et al., 2022; Petersen et al., 2021, 2022). However, in the context of sparse MoEs, it is crucial that only k experts process a given token x. On the negative side, the aforementioned operators are dense, in contrast to the original top-k, which is sparse. In this manuscript (Chapter 7), we present a sparse and differentiable everywhere relaxation for the top-k operator and validate its benefits experimentally at scale.

1.1.8 In this manuscript

Now that we have introduced the main tools and challenges we are interested in this manuscript, we will present a detailed overview of our contributions in sections 1.2, 1.3 and 1.4. Each standalone contribution is then presented in the rest of the manuscript. We also emphasize that to every chapter of this manuscript corresponds an open-source python code to reproduce the experimental findings or use our proposed models. We provide the corresponding repositories at the end of each chapter overview.

1.2 Mathematical foundations for the connection between Residual Neural Networks and Neural Ordinary Differential Equations [Part I of the manuscript]

We saw how residual neural networks (He et al., 2016a,b) keep on outperforming state of the art in computer vision (Wightman et al., 2021; Bello et al., 2021), and more generally skip connections are widely used in a various range of applications (Vaswani et al., 2017; Dosovitskiy et al., 2020). This part of the manuscript is dedicated to formally study the connection between residual networks and neural ODEs. Therefore, we consider a simple modification of the residual network forward rule (1.4) by letting explicitly the residual mapping depend on the depth N of the network:

$$x_{n+1} = x_n + \frac{1}{N} f(x_n, \theta_n^N). \tag{1.9}$$

Note that this correspond to a particular instance of Stable Resnets introduced in Hayou et al. (2021). On the other hand, we consider the Neural ODE (1.6):

$$\frac{\mathrm{d}X}{\mathrm{d}s} = f(X(s), \Theta(s)) \quad \text{with} \quad X(0) = x_0.$$

which outputs a final value $x(1) \in \mathbb{R}^d$, the solution of equation (1.6).

Now consider the Euler scheme for solving equation (1.6) with time step $\frac{1}{N}$ starting from x_0 and iterating $x_{n+1} = x_n + \frac{1}{N} f(x_n, \Theta(\frac{n}{N}))$. Under mild assumptions on f and Θ, this scheme is known to converge to the true solution of equation (1.6) as N goes to $+\infty$. If in addition $\Theta(\frac{n}{N}) = \theta_n^N$, then the ResNet equation (1.9) corresponds to a Euler discretization with time step $\frac{1}{N}$ of equation (1.6). In fact, we have the following result.

Proposition 1.1. *Suppose there exists a smooth function $\Theta : [0,1] \to \mathbb{R}^k$ such that $\theta_n^N = \Theta(\frac{n}{N})$ for all N and f is Lipschitz continuous on compact sets. Then the Neural ODE (1.6) has a unique solution and $\|X(\frac{n}{N}) - x_n\| = O(\frac{1}{N})$. In particular, $x_N \to X(1)$ as the number of layers $N \to \infty$.*

However, for a given ResNet with fixed depth and weights, the activations in equation (1.9) can be far from the solution of equation (1.6). This is illustrated in Figure 1.9 where we show that a deep ResNet can easily break the topology of the input space, which is impossible for a first order ODE.

These different behaviors require a mathematical framework to precisely characterize the conditions under which deep residual networks behave like neural ODEs, which we propose in Chapters 2 and 3. The potential ODE structure for the trained network is significant for several reasons. First, it clarifies the connection between trained residual networks and neural ODEs, reinforcing the common statement that both coincide in the large-depth limit (see, e.g., Haber and Ruthotto, 2017a; E et al., 2019; Dong et al., 2020; Massaroli et al., 2020; Kidger, 2022). Secondly, if trained

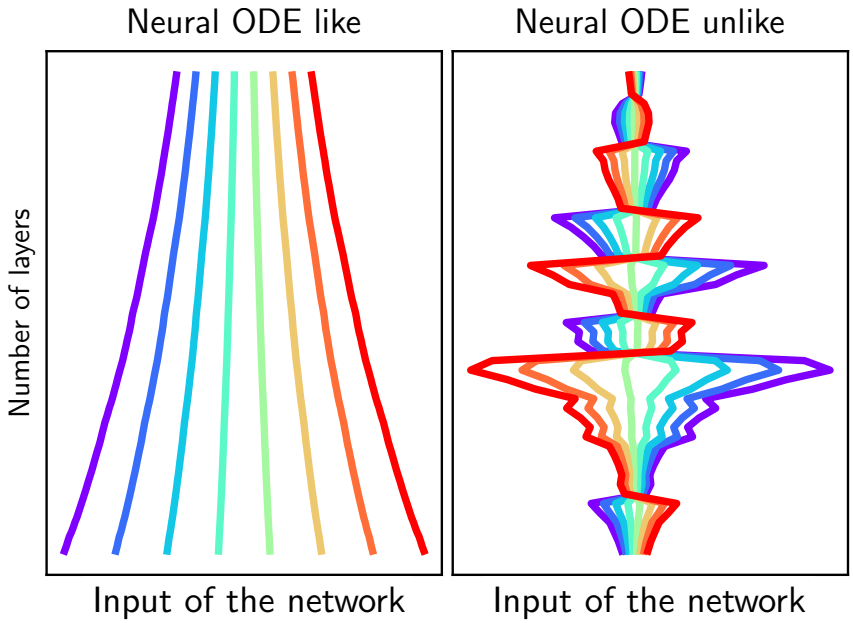

Figure 1.9: **Trajectory of ResNets with 300 layers.** Left: we learn $x \to \frac{x}{2}$, trajectories are smooth and do not intersect. Right: we learn $x \to \frac{-x}{2}$, trajectories are not smooth and intersect.

residual networks are discretizations of neural ODEs, results from neural ODEs can be applied to understand residual networks better, offering theoretical insights into approximation capabilities (Teshima et al., 2020; Zhang et al., 2020a) and practical benefits like memory-efficient training (Chen et al., 2018; Sander et al., 2022b) and weight compression (Queiruga et al., 2021). Finally, it provides a step towards understanding the implicit regularization (Neyshabur et al., 2014; Vardi, 2023) of gradient descent in deep residual networks.

Recent studies have examined the connection between residual networks and neural ODEs. In Cohen et al. (2021), experiments are conducted to better understand how the weights in ResNets scale with network depth. The authors show that under the assumption of a scaling limit $\Theta(s) = N^\beta \lim \theta^N_{\lfloor Ns \rfloor}$ for the weights (where $0 < \beta < 1$) and a network scaling factor of $\frac{1}{N^\alpha}$ (with $0 < \alpha < 1$ and $\alpha + \beta = 1$), the hidden state of the network converges to a solution of a linear ODE. In Chapters 2 and 3, we focus on the case where $\alpha = 1$, as it corresponds to the scaling in Euler's method with step size $\frac{1}{N}$. The recent study by Cont et al. (2022) investigates the linear convergence of gradient descent in residual networks, proving the existence of a $\frac{1}{2}$-Hölder continuous scaling limit as $N \to \infty$, with a scaling factor for the residuals of $\frac{1}{\sqrt{N}}$, which contrasts with our $\frac{1}{N}$ scaling. In contrast, we show that our limit function is Lipschitz continuous, indicating stronger regularity. This finding aligns with the work of Marion et al. (2022), which shows that while $\frac{1}{\sqrt{N}}$ scaling is suitable for standard i.i.d. initializations, $\frac{1}{N}$ is the correct scaling for smooth initialization to achieve non-trivial behavior (other choices lead to explosion or identity (Marion et al., 2022)). More broadly, recent studies have shown the convergence of gradient descent in residual networks when the initial loss is sufficiently small. This includes residual networks with finite width but arbitrary depth (Du et al., 2019; Liu et al., 2020) and networks with both infinite width and depth (Lu et al., 2020; Barboni et al., 2021). These proofs rely on an implicit bias towards weights with small amplitudes, but the question of the convergence of individual weights as depth increases remains open, and we address it in this work. This requires demonstrating an additional bias towards weights with small variations across depth.

1.2.1 Do Residual Neural Networks Discretize Neural Ordinary Differential Equations? [Chapter 2 of the manuscript]

We begin to investigate the connection between residual networks and Neural ODEs in Chapter 2. Our main contributions are to quantify how closely the discrete dynamics of a residual network approximate the continuous dynamics of a Neural ODE and to propose a discrete approximate adjoint method to train a residual network without the memory consumption needed to store the activations.

Approximation error: closeness to an ODE. We propose a framework to define an associated Neural ODE to a given residual network and control the error between discrete and continuous trajectories: we establish a bound on this distance, summarized in the following proposition.

Proposition 1.2 (Approximation error)**.** *Let $\varphi_\Theta : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ such that $\varphi_\Theta(x, s) = f(x, \Theta(s))$. Suppose that $\forall n \in \{0, ..., N-1\}$, $\varphi_\Theta(., \frac{n}{N}) = f(., \theta_n^N)$, that φ_Θ is $\mathcal{C}^1$, and L-Lipschitz with respect to x, uniformly in s. Note that this implies that the solution of equation (1.6) is well defined, unique, $\mathcal{C}^2$, and that the trajectory is included in some compact $K \subset \mathbb{R}^d$. Denote $C_N := \|\partial_s \varphi_\Theta + \partial_x \varphi_\Theta[\varphi_\Theta]\|_\infty^{K \times [0,1]}$. Then one has for all n: $\|x_n - x(\frac{n}{N})\| \leq \frac{e^L - 1}{2NL} C_N$.*

Using this result, we find on the negative side that if the residual functions are not smooth with depth, this bound does not diminish as the depth N increases.

On the positive side, we show that gradient descent preserves the smoothness of residual functions for a residual network with linear residuals and sufficiently small initial loss, leading to an implicit regularization towards a limit Neural ODE at a rate of $\frac{1}{N}$, uniformly with respect to both depth and optimization time. More precisely, given a training set $(x^i, y^i)_{i \in [p]}$ in $\mathbb{R}^d$, we solve the regression problem of mapping x^i to y^i with a linear ResNet, *i.e.* $f(x, \theta) = \theta x$, of depth N and parameters $(\theta_1^N, \ldots, \theta_N^N)$. It corresponds to a deep matrix factorization problem (Zou et al., 2020a; Bartlett et al., 2018; Arora et al., 2019, 2018). As opposed to these previous works, we study the infinite depth limit of these linear ResNets with a focus on the learned weights with gradient flow (1.2). Our result takes the following informal form.

Proposition 1.3 (Smoothness in depth of the weights)**.** *Suppose that the initial loss is sufficiently small and that $\|\theta_n^N(0)\| \leq \frac{1}{4}$. Suppose that there exists $C_0 > 0$ independent of n and N such that $\|\theta_{n+1}^N(0) - \theta_n^N(0)\| \leq \frac{C_0}{N}$. Then, $\forall t \in \mathbb{R}_+$, $\|\theta_n^N(t)\| < \frac{1}{2}$, and $\theta_n^N(t)$ admits a limit ψ_n^N as $t \to +\infty$. Moreover, there exists $C > 0$ such that $\forall t \in \mathbb{R}_+$, $\|\theta_{n+1}^N(t) - \theta_n^N(t)\| \leq \frac{C}{N}$.*

We show that a consequence of this result is an implicit regularization of the linear residual network towards a linear neural ODE at speed $\frac{1}{N}$.

Memory-free discrete adjoint method. Additionally, we explore the use of a memory-free discrete adjoint method to train ResNets by recovering activations on-the-fly during the backward pass, hence avoiding the need for storing activations during the forward pass (Wang et al., 2018; Peng et al., 2017; Zhu et al., 2017; Gomez et al., 2017). These memory savings come at the price of an about 1.5 times slower training: we are trading memory for computations. Our method corresponds to define $\tilde{x}_N = x_N$ and iterate for $n \in \{N-1, \ldots 0\}$:

$$\tilde{x}_n = \tilde{x}_{n+1} - \frac{1}{N} f(\tilde{x}_{n+1}, \theta_n^N).$$

We propose to modify the backpropagation equations (1.5) by using the approximated activations $(\tilde{x}_n)_{n \in [N]}$ as a proxy for the true activations $(x_n)_{n \in [N]}$ to compute gradients without storing the

activations:

$$\tilde{\nabla}_{\theta_{n-1}^N}\ell = \frac{1}{N}[\partial_\theta f(\tilde{x}_{n-1}, \theta_{n-1}^N)]^\top \nabla_{\tilde{x}_n}\ell, \quad \nabla_{\tilde{x}_{n-1}}\ell = [I + \frac{1}{N}\partial_x f(\tilde{x}_{n-1}, \theta_{n-1}^N)]^\top \nabla_{\tilde{x}_n}\ell.$$

Contrarily to other models such as RevNets (Gomez et al., 2017) (architecture change) or Momentum ResNets presented in Chapter 4 (forward rule modification) which rely on an exactly invertible forward rule, the proposed method requires no change at all in the network, but gives approximate gradients. We show that this method is theoretically sound at large depths if the residual functions are Lipschitz continuous with respect to the input. Furthermore, we demonstrate that Heun's method, a second-order ODE integration scheme, provides better gradient estimates with the adjoint method when the residual functions are smooth with depth.

Our experimental validation supports these findings: the adjoint method is successful at large depths, and Heun's method reduces the number of layers required for effective training. We also apply the adjoint method for fine-tuning very deep ResNets without additional memory consumption in the residual layers.

Software. Our code is available at `https://github.com/michaelsdr/resnet_nodes`.

1.2.2 Implicit regularization of deep residual networks towards neural ODEs [Chapter 3 of the manuscript]

In Chapter 3, we significantly extend the result of Proposition 1.2.1 regarding the ODE structure of trained residual networks by considering non-linear models. Specifically, we examine the formulation

$$x_{n+1} = x_n + \frac{1}{N\sqrt{m}} V_{n+1}^N \sigma\Big(\frac{1}{\sqrt{q}} W_{n+1}^N x_n\Big), \quad n \in \{0, \dots, N-1\}, \tag{1.10}$$

where N is the network depth, $V_n^N \in \mathbb{R}^{d\times m}$, $W_n^N \in \mathbb{R}^{m\times d}$ are layer weights, and σ is an element-wise activation function. The scaling factors are made explicit to maintain weights of magnitude $\mathcal{O}(1)$ regardless of width and depth. As in the previous section, the $1/N$ scaling factor is essential for the correspondence with neural ODEs. As in Proposition 1.2, if Lipschitz continuous functions $\mathcal{V}$ and $\mathcal{W}$ exist such that $V_n^N = \mathcal{V}(n/N)$ and $W_n^N = \mathcal{W}(n/N)$, the residual network (1.10) converges, as $N \to \infty$, to the ODE:

$$\frac{dX}{ds}(s) = \frac{1}{\sqrt{m}} \mathcal{V}(s)\sigma\Big(\frac{1}{\sqrt{q}} \mathcal{W}(s)X(s)\Big), \quad s \in [0,1], \tag{1.11}$$

This correspondence holds for fixed limiting functions $\mathcal{V}$ and $\mathcal{W}$ at initialization. However, similarly to the linear case presented in 1.2.1, the structure of the network during and after training, when weights are updated with gradient descent, is more complex and requires further analysis.

We assume that the network is trained with rescaled version of the gradient flow (1.2), where the parameters V_n are updated according to the ODE $\frac{dV_n}{dt}(t) = -N\frac{\partial \ell}{\partial V_n}(t)$ for $t \geq 0$, and similarly for W_n. The scaling factor N prevents vanishing gradients as N increases, and the gradient flow is defined with respect to a time index t distinct from the layer index s.

Main contributions. Our first main contribution is to show in Theorem 3.4 that a neural ODE limit holds after training up to time t, i.e., there exists a function $\mathcal{V}(s,t)$ such that the residual network converges, as N tends to infinity, to the ODE:

$$\frac{dX}{ds}(s) = \frac{1}{\sqrt{m}}\mathcal{V}(s,t)\sigma\Big(\frac{1}{\sqrt{q}}\mathcal{W}(s,t)X(s)\Big), \quad s \in [0,1].$$

This large-depth limit holds for any finite training time $t \geq 0$. However, convergence of the optimization algorithm as t tends to infinity, referred to as the long-time limit, is not guaranteed without further assumptions due to the non-convexity of the optimization problem. We address this by proving a Polyak-Łojasiewicz (PL) condition, which ensures long-time convergence of the gradient flow. This condition implies that the gradients of the residual network's loss cannot be small when the loss itself is large, and can be seen as a relaxation of convexity (see Definition 3.5). We prove this condition for sufficiently wide networks of the form (1.10) in Proposition 3.6. Finally, we show in Theorem 3.7 with high probability that as both N and t tend to infinity, the trained residual network (1.10) converges to the solution of the neural ODE (1.11), with limiting functions $\mathcal{V}_\infty$ and $\mathcal{W}_\infty$.

Generalizations to other architecture. As shown in Theorem 3.15 and Proposition 3.16, most of our results actually hold for a more general residual network of the form

$$x_{n+1} = x_n + \frac{1}{N} f(x_n, \theta_n^N), \tag{1.12}$$

where $f : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^d$ is a $\mathcal{C}^2$ function such that $f(0,\cdot) \equiv 0$ and $f(\cdot,\theta)$ is uniformly Lipschitz for θ in any compact. However, for this generalized model, we need to assume the existence of a PL condition to prove the convergence in the long-time and large-depth limits.

Our network of interest (1.10) is a special case of model (1.12), and other choices include convolutional layers or a Lipschitz continuous version of Transformer (Kim et al., 2021), that is using a L_2 cost in equation (1.8). This latter case is particularly interesting in the light of the literature analyzing Transformers from a neural ODE point of view (Lu et al., 2019; Sander et al., 2022a; Geshkovski et al., 2023).

Experiments. Our results are illustrated by numerical experiments where we train convolutional ResNets on CIFAR and find out that the smoothness of convolutional layers is preserved during training, in contrast to i.i.d initialization, as shown in Figure 1.10.

Software. Our code is available at `https://github.com/michaelsdr/implicit-regularization-resnets-nodes`.

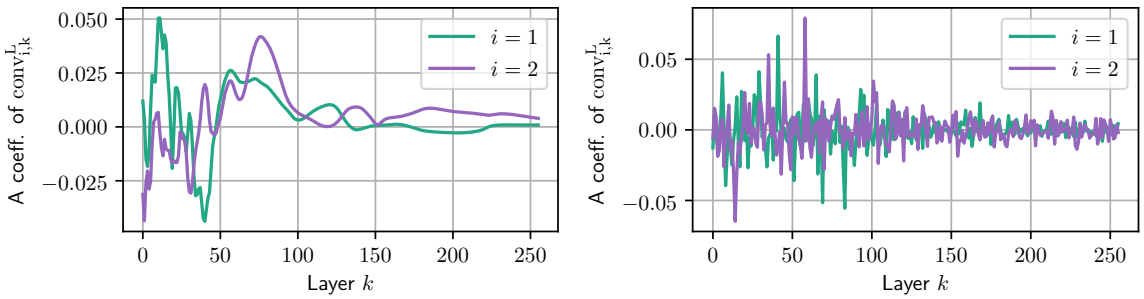

Figure 1.10: Random entries of the convolutions across layers (x-axis) after training. **Left:** Weight-tied initialization leads to smooth weights. **Right:** i.i.d. initialization leads to non-smooth weights.

1.3 Analogy between Residual Neural Networks and Neural Ordinary Differential Equations to design and study new architectures [Part II of the manuscript]

In the second part of the manuscript, we introduce two new architectures and leverage the analogy with neural ODEs to explore their properties. The first model, called Momentum ResNet, serves as a drop-in replacement for any residual network and is designed to reduce memory usage. The second model, termed Sinkformer, is a drop-in replacement for any Transformer architecture. It involves replacing the row-wise stochastic attention matrices typically used in Transformers with doubly stochastic attention matrices, achieved through the application of Sinkhorn algorithm.

1.3.1 Momentum Residual Neural Networks [Chapter 4 of the manuscript]

We saw that training deep residual neural networks with backpropagation incurs a memory cost that scales linearly with network depth and batch size. This is a practical bottleneck, particularly when using graphics processing units (GPUs) (Wang et al., 2018). To address this, we propose, in Chapter 4, Momentum Residual Neural Networks (Momentum ResNets), which introduce a momentum term into the ResNet forward rule. More precisely, given the standard residual network's forward rule (1.4), we define the Momentum ResNet forward rule as:

$$\begin{cases} v_{n+1} = \gamma v_n + (1-\gamma) f(x_n, \theta_n), \\ x_{n+1} = x_n + v_{n+1}, \end{cases}$$

where v_n is a velocity term and $\gamma \in [0, 1]$ is a momentum term.

Invertibility. This modification changes the network's dynamics, but most importantly, it makes Momentum ResNets invertible. Indeed, the activations at layer n can be recovered from those at layer $n+1$ using the following equations:

$$\begin{cases} x_n = x_{n+1} - v_{n+1}, \\ v_n = \frac{1}{\gamma}\left(v_{n+1} - (1-\gamma) f(x_n, \theta_n)\right). \end{cases}$$

This invertibility allows for on-the-fly reconstruction of activations during the backward pass, leading to substantial memory savings. The memory cost is reduced from $O(k \times d \times n_{\text{batch}})$ to $O((1-\gamma) \times k \times d \times n_{\text{batch}})$, where k is the network depth, d is the dimensionality of the activations, and n_{batch} is the batch size. Similarly to the discrete adjoint method presented in section 1.2.1, this memory savings comes a the price of an about 1.5 times slower training: we are trading memory for computations.

Drop-in replacement. Unlike previous invertible models (Gomez et al., 2017; Behrmann et al., 2019; Chang et al., 2018), Momentum ResNets can serve as a **drop-in replacement** for any existing ResNet block without altering the overall architecture. In practice, one can define the Momentum ResNet counterpart of any existing ResNet in one line of code.

Continuous Limit Interpretation. In the limit of infinitesimal step size, Momentum ResNets can be interpreted as discretizations of second-order ODEs:

$$\varepsilon \ddot{x} + \dot{x} = f(x, \theta), \tag{1.13}$$

where $\varepsilon = \frac{1}{1-\gamma}$. This contrasts with traditional ResNets, which correspond to first-order ODEs (Chen et al., 2018).

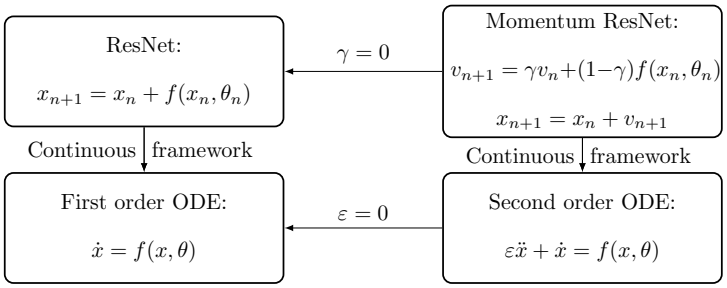

Figure 1.11: **Overview of the four different paradigms.**

Representation Capabilities. The introduction of the momentum term gives Momentum ResNets the ability to represent a broader class of functions. It is well-known that a single residual block has universal approximation capabilities (Cybenko, 1989), allowing any continuous function on a compact set to be uniformly approximated by a one-layer feedforward network. However, neural ODEs have limited representation capabilities (Teh et al., 2019; Li et al., 2019; Teshima et al., 2020; Zhang et al., 2020b). We demonstrate that our architecture has superior representation capabilities in the continuous limit (Proposition 4.2). We also prove its universality in the linear case in Theorem 4.4, proving they can represent any linear mapping up to a multiplicative factor, whereas traditional ResNets cannot.

Experiments. We evaluated Momentum ResNets on CIFAR-10, CIFAR-100 (Krizhevsky et al., 2010), and ImageNet (Deng et al., 2009b). Our results show that Momentum ResNets achieve comparable accuracy to ResNets while significantly reducing memory usage.

Software. Our code is available at `https://github.com/michaelsdr/momentumnet`.

1.3.2 Sinkformers: Transformers with Doubly Stochastic Attention [Chapter 5 of the manuscript]

In Chapter 5, our contributions are two folds. On the one hand, we introduce a new variant of the Transformer called Sinkformer by considering doubly stochastic attention matrices. Sinkformers are motivated by the empirical evidence that, across various modalities, attention matrices in Transformers get closer to doubly-stochastic matrices during training. On the other hand, we show that Transformers and Sinkformers can be viewed as models acting on probability measures and we study the corresponding infinite depth limit models, showing they correspond to neural PDEs on the space of probability measures, and to Wasserstein gradient-flows for Sinkformers.

Sinkformers. First, we extend the Transformer architecture (Vaswani et al., 2017), which is known for its empirical success in NLP (Brown et al., 2020; Radford et al., 2019; Wolf et al., 2019) and computer vision (Dosovitskiy et al., 2020; Zhao et al., 2020; Zhai et al., 2021; Lee et al., 2019a), by introducing a variant called Sinkformer. The only modification in Sinkformers is replacing the SoftMax operator in the attention mechanism with Sinkhorn's algorithm (Sinkhorn, 1964; Cuturi, 2013; Peyré et al., 2019), which normalizes the attention matrix to be doubly stochastic. Intuitively, such a normalization relies on a democratic principle where all points

are matched one to another with different degrees of intensity so that more interactions are considered than with the SoftMax normalization, as shown in Figure 1.12.

For simplicity, we focus on a one-head standard self-attention mechanism (taking $H = 1$ in equation (1.8)), which we recall operates on a sequence $x := (\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_T)$ embedded in dimension d:

$$\mathrm{x}_t \leftarrow \mathrm{x}_t + \sum_{t'=1}^{T} \mathcal{A}^1_{t,t'} W_V \mathrm{x}_{t'},$$

where the attention matrix $\mathcal{A}^1$ is given by $\mathcal{A}^1_{t,:} = \mathrm{SoftMax}(C_{t,:})$ with $C_{t,t'} = (W_Q \mathrm{x}_t)^\top W_K \mathrm{x}_{t'}$. Here, $W_Q, W_K \in \mathbb{R}^{m \times d}$ and $W_V \in \mathbb{R}^{d \times d}$ are the query, key, and value matrices, respectively. The SoftMax normalization ensures that $\mathcal{A}^1$ is a row-stochastic matrix.

We propose to extend this normalization by applying Sinkhorn's algorithm, which alternates row and column normalizations of $\mathcal{A}^0 = \exp(C)$ until convergence to a doubly stochastic matrix, denoted as $\mathcal{A}^\infty$. The resulting Transformer variant, the Sinkformer, can be seen as interpolating between a classical Transformer and a model that fully leverages the Sinkhorn normalization process (Cuturi, 2013; Peyré et al., 2019).

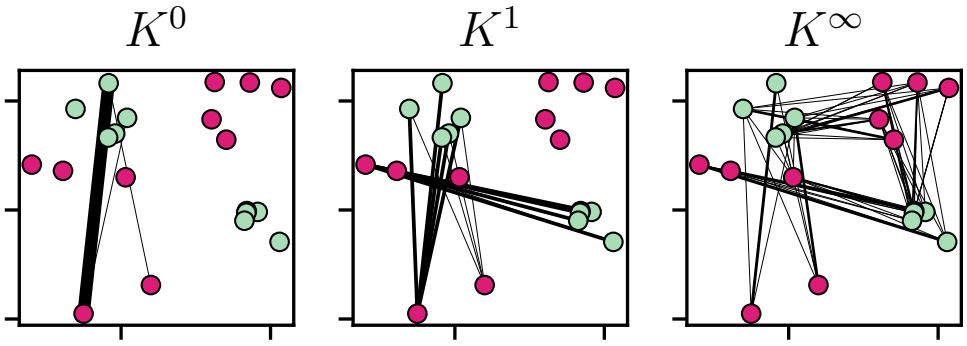

Figure 1.12: **Illustration of the different normalizations of attention matrices.** We form two point clouds $(W_Q x_t)_{1 \le t \le 10}$ (green) and $(W_K x_t)_{1 \le t \le 10}$ (red). For $k \in \{0, 1, \infty\}$, the width of the line connecting x_t to $x_{t'}$ is $\mathcal{A}^k_{i,j}$. We only display connections with $\mathcal{A}^k_{i,j} \ge 10^{-12}$. For $\mathcal{A}^0$, one interaction dominates. For $\mathcal{A}^1$ (SoftMax), one cluster is ignored. For $\mathcal{A}^\infty$ (Sinkhorn), all points are involved in an interaction.

Transformers as flows on the space of probability measures. Our theoretical contributions include showing that Transformers and Sinkformers can be interpreted as models acting on probability distributions. Namely, we consider the measure framework presented in Section 1.1.6. We denote $c(x, x') := (W_Q x)^\top W_K x'$ and $a^0 := \exp(c)$. For some measure $\mu \in \mathcal{M}(\mathbb{R}^d)$, we denote $a^1(x, x') := \mathtt{SoftMax}(c)(x, x') = \frac{a^0(x,x')}{\int a^0(x,y) d\mu(y)}$ and similarly, we denote $a^\infty := \mathtt{Sinkhorn}(c)$. Note that a^0, a^1 and a^∞ are the continuous equivalent of the matrices $\mathcal{A}^0$, $\mathcal{A}^1$ and $\mathcal{A}^\infty$ respectively. We also formalize the neural ODE (1.6) in this context: we show that an infinite succession of attention blocks with tied weights between layers defines the neural PDE

$$\partial_t \mu + \mathrm{div}(\mu \Gamma^k_\mu) = 0,$$

with $\Gamma^k_\mu(x) := \int a^k(x, x') W_V x' d\mu(x')$.

Under a symmetry assumption, we demonstrate that Sinkformers, in the infinite depth limit, can be viewed as Wasserstein gradient flows (Santambrogio, 2017) for an energy minimization problem. Specifically, we establish the following proposition.

Proposition 1.4 (PDEs associated to a^0, a^1, a^∞). *Suppose that* ${W_K}^\top W_Q = {W_Q}^\top W_K = -W_V$. *Let* $\mathcal{F}^0$ *and* $\mathcal{F}^\infty$ *be functionals on the space of probability measures* $\mathcal{M}(\mathbb{R}^d)$ *defined by:*

$$\mathcal{F}^0(\mu) = \frac{1}{2}\int a^0(\mathrm{x},\mathrm{x}')\, d(\mu\otimes\mu), \quad \textit{and} \quad \mathcal{F}^\infty(\mu) = -\frac{1}{2}\int a^\infty(\mathrm{x},\mathrm{x}')\log\left(\frac{a^\infty(\mathrm{x},\mathrm{x}')}{a^0(\mathrm{x},\mathrm{x}')}\right) d(\mu\otimes\mu).$$

Then, the dynamics of an infinite number of Sinkformer and Transformer attention layers are governed by the following PDEs:

$$\frac{\partial\mu}{\partial t} + div(\mu\Gamma^k_\mu) = 0, \quad \textit{with } \Gamma^k_\mu(\mathrm{x}) = \begin{cases} -\nabla_W\mathcal{F}^0(\mu)(\mathrm{x}) & \textit{if } k=0,\\ -\nabla\left[\log\left(\int a^0(\mathrm{x},\mathrm{x}')\, d\mu(\mathrm{x}')\right)\right] & \textit{if } k=1,\\ -\nabla_W\mathcal{F}^\infty(\mu)(\mathrm{x}) & \textit{if } k=\infty. \end{cases}$$

In addition, $\Gamma^1_\mu(\mathrm{x}) = -\nabla\left[\log\left(\int a^0(\mathrm{x},\mathrm{x}')\, d\mu(\mathrm{x}')\right)\right]$ *is not a Wasserstein gradient.*

Experiments. On the experimental side, we find out that Sinkformers lead to accuracy gains across various tasks. For instance, on the ModelNet 40 3D shape classification task, Sinkformers outperform classical Transformers in terms of both mean and median accuracy (Wu et al., 2015; Guo et al., 2021). Similarly, in sentiment analysis on the IMDb dataset (Maas et al., 2011) and neural machine translation on the IWSLT'14 German to English dataset (Cettolo et al., 2014), Sinkformers consistently achieve better performance, highlighting the practical benefits of the proposed method.

Software. Our code is available at `https://github.com/michaelsdr/sinkformers`.

1.4 Transformers in Action [Part III of the manuscript]

In the third and last part of the manuscript, we explore additional insights into Transformers. First, building on the works presented in Section 1.1.7.1, we study in Section 1.4.1 how Transformers perform next-token prediction in an autoregressive manner. Secondly, we consider the sparse mixture of experts framework for Transformers, introduced in Section 1.1.7.2 and show in Section 1.4.2 how tokens can be routed to experts in a differentiable manner. To achieve this, we introduce new sparse and differentiable top-k operators.

1.4.1 How do Transformers perform In-Context Autoregressive Learning? [Chapter 6 of the manuscript]

Causal Transformers are trained to predict the next token s_{T+1} given a sequence (also termed as context) $s_{1:T} \coloneqq (s_1, \cdots, s_T)$. An intriguing property of large Transformers is their ability to adapt their computations given the context $s_{1:T}$. In Chapter 6, we make a step towards understanding this *in-context learning* ability. More precisely, building on the work of Von Oswald et al. (2023b), we focus on a simple autoregressive (AR) process of order 1, where each sequence is generated following the recursion $s_{T+1} = \varphi_W(s_{1:T}) \coloneqq Ws_T$, and W is a randomly sampled orthogonal matrix, referred to as the *context matrix*. We say that a trained Transformer *autoregressively learns* this relation *in-context* if it satisfies the following definition.

Definition 1.5 (In-Context Autoregressive Learning). *A trained Transformer is said to autoregressively learn in-context the AR process* $s_{t+1} = Ws_t$ *if it decomposes its prediction into two steps: (1) estimating* W *through an in-context mapping, and (2) applying a simple prediction mapping* ψ*, which is equal or closely related to* φ_W.

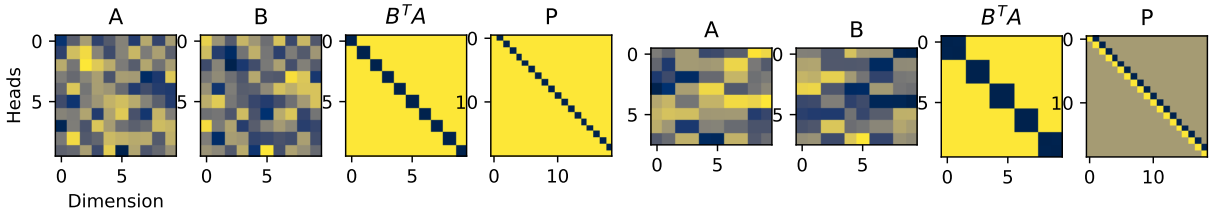

Figure 1.13: **Matrices $\mathtt{A}$, $\mathtt{B}$, $\mathtt{B}^\top\mathtt{A}$, and P after training a linear Transformer model on the loss $\ell(\theta)$ with random initialization**. We take $d = 10$ and $T = 15$. **Left:** Unitary context case with $H = 10$. **Right:** Orthogonal context case, with $H = 8 < d$, leading to low rank $\mathtt{B}^\top\mathtt{A}$. In both cases, we obtain arbitrarily small final loss and recover parameters exhibiting structure.

Our goal is to fully characterize the autoregressive in-context learning process for optimally-trained Transformers. We consider two settings, namely the augmented and non-augmented settings, in which the tokens are first mapped to new tokens $\mathrm{x}_{1:T}$.

Model and objective. We consider a variant of equation (1.8) by considering a model $\mathcal{T}_\theta$ involving Causal Linear Multi-Head Attention:

$$\mathrm{x}_{1:T} \mapsto \left(\sum_{h=1}^{H} \sum_{t'=1}^{t} \mathcal{A}^h_{t,t'} B^h \mathrm{x}_{t'} \right)_{t \in \{1,\cdots,T\}}. \tag{1.14}$$

$\mathcal{A}^h$ is the attention matrix: $\mathcal{A}^h_{t,t'} = P_{t,t'} \langle A^h \mathrm{x}_t \mid \mathrm{x}_{t'} \rangle$, with $P \in \mathbb{R}^{T_{\max} \times T_{\max}}$ an optionally trainable positional encoding. The trainable parameters are $\theta = \big((A^h, B^h)_{1 \le h \le H}, P\big)$.

We focus on the population loss indicating the model's objective to predict s_{T+1} given $\mathrm{x}_{1:T}$, defined as $\ell(\theta) := \sum_{T=2}^{T_{\max}} \mathbb{E}_{W \sim \mathcal{W}} \|\mathcal{T}_\theta(\mathrm{x}_{1:T}) - s_{T+1}\|^2$.

Augmented Setting: In the augmented setting, we consider tokens defined as $\mathrm{x}_t := (0, s_t, s_{t-1})$, aligning with the setup used by Von Oswald et al. (2023b). We consider the model $\mathcal{T}_\theta(\mathrm{x}_{1:T}) = \left(\mathrm{x}_T + \sum_{t=1}^{T} \langle A \mathrm{x}_T | \mathrm{x}_t \rangle B \mathrm{x}_t \right)_{1:d}$. We show that this model trained at optimality performs one step of gradient descent on an inner objective as its in-context mapping. The result takes the following informal form.

Proposition 1.6 (In-Context Autoregressive Learning with Gradient-Descent)**.** *Suppose the context matrices W are unitary (i.e., $W \in U(d)$) and commute, and the model parameters are initialized with a block structure. Then, when the loss $\ell(\theta)$ is minimized, the optimal in-context mapping Γ_{θ^*} corresponds to one step of gradient on the loss $L(W, \mathrm{x}_{1:T}) = \frac{1}{2} \sum_{t=1}^{T-1} \|s_{t+1} - W s_t\|^2$ starting from the initialization $W = 0$, with a step size asymptotically equivalent to $\frac{3}{2T_{\max}}$ as $T_{\max}$ increases.*

Non-Augmented Setting: In the non-augmented setting, we consider tokens $\mathrm{x}_t := s_t$ and study a multi-head self-attention model defined as:

$$\mathcal{T}_\theta(\mathrm{x}_{1:T}) = \sum_{h=1}^{H} \sum_{t=1}^{T} P_{T-1,t} \langle \mathrm{x}_t | A^h \mathrm{x}_{T-1} \rangle B^h \mathrm{x}_t, \tag{1.15}$$

where $\mathcal{T}_\theta$ is a linear Transformer with positional encoding. We suppose that $A^h := \mathrm{diag}(a_h)$ and $B^h := \mathrm{diag}(b_h)$ are diagonal and define $\mathtt{A} := (a^1, \cdots, a^d) \in \mathbb{R}^{H \times d}$, and $\mathtt{B} := (b^1, \cdots, b^d)$. Our results in Propositions 6.7 and 6.11 show that the trained Transformer learns an in-context mapping and prediction mapping by leveraging positional encodings and head-wise orthogonality, which is emphasized in Figure 1.13.

Experiments. We also conduct experiments to generalize our theoretical findings when the matrices W do not commute:

- *Augmented Setting:* We find that a single-step of gradient descent matches optimal Transformer performance, but additional layers don't improve beyond this baseline, suggesting that deeper models need extra mechanisms.
- *Non-Augmented Setting:* Our experiments confirm theoretical predictions, showing orthogonality across attention heads and highlighting the significant role of positional encodings in learning geometric token relationships.
- *Context Distribution:* Varying context matrix distributions alters learned positional encodings and in-context mappings, indicating the influence of optimization on Transformer behavior in autoregressive tasks.

Software. Our code is available at `https://github.com/michaelsdr/ical`.

1.4.2 Fast, differentiable and sparse top-k: a convex analysis perspective [Chapter 7 of the manuscript]

We saw in Section 1.1.7.2 how finding the top-k values and their corresponding indices in a vector is an essential component in Transformer-based sparse mixture of experts (MoEs) (Shazeer et al., 2017; Fedus et al., 2022) (see Figure 1.8), where the top-k router assigns each token to a subset of k experts, or alternatively, maps each expert to a subset of k tokens.

However, the top-k operator is inherently a discontinuous piecewise affine function, with derivatives that are either undefined or constant. This characteristic poses challenges when using it in neural networks that are trained via gradient backpropagation. Recent approaches have sought to address this by introducing differentiable relaxations (Amos et al., 2019; Qian et al., 2022; Petersen et al., 2021, 2022). Despite these advancements, no current method achieves both differentiability everywhere and sparsity.

Sparsity is particularly crucial in neural networks requiring conditional computation, such as in sparse mixtures of experts. Without sparsity, every expert would need to process all tokens, significantly increasing computational costs. We propose new differentiable *and* sparse top-k operators. Our framework is significantly more general than the existing one and allows for example to express top-k operators that select values *in magnitude*. On the algorithmic side, in addition to pool adjacent violator (PAV) algorithms, we propose a new GPU/TPU-friendly Dykstra algorithm to solve isotonic optimization problems.

Building a differentiable Top-k Operator Our approach is grounded in the framework introduced by Blondel et al. (2020b), which formulates sorting and ranking as linear programs over the permutahedron. For any mapping $\varphi(\boldsymbol{x}) := (\varphi(x_1), \dots, \varphi(x_n))$, we define

$$f_\varphi(\boldsymbol{x}, \boldsymbol{w}) := f(\varphi(\boldsymbol{x}), \boldsymbol{w}) \quad \text{and} \quad \mathbf{y}_\varphi(\boldsymbol{x}, \boldsymbol{w}) := \nabla_1 f_\varphi(\boldsymbol{x}, \boldsymbol{w})$$

with

$$f(\boldsymbol{x}, \boldsymbol{w}) := \max_{\boldsymbol{y} \in P(\boldsymbol{w})} \langle \boldsymbol{x}, \boldsymbol{y} \rangle \quad \text{and} \quad \mathbf{y}(\boldsymbol{x}, \boldsymbol{w}) := \operatorname*{argmax}_{\boldsymbol{y} \in P(\boldsymbol{w})} \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \nabla_1 f(\boldsymbol{x}, \boldsymbol{w}),$$

where $P(\boldsymbol{w}) \coloneqq \operatorname{conv}(\{\boldsymbol{w}_\sigma \colon \sigma \in \Sigma\}) \subset \mathbb{R}^n$ is the permutahedron associated to a vector $\boldsymbol{w}$. It is easily seen that using $\varphi(x) = x$ leads to a top-k mask operator and using $\varphi(x) = \frac{1}{2}x^2$ leads to a top-k in magnitude. We then introduce a p-norm regularization term $R(\mathbf{y}) = \frac{1}{p}\|\mathbf{y}\|_p^p$ by defining

$f^*_{\varphi,R}(\mathbf{y}, \boldsymbol{w}) := f^*_\varphi(\mathbf{y}, \boldsymbol{w}) + R(\mathbf{y})$. This regularization induces smoothness in the operator, enabling gradients to be computed efficiently during backpropagation. For instance, with $p = 4/3$, the regularized operator becomes continuously differentiable.

Going back to the primal, we show in Proposition 7.2 that

$$f_{\varphi,R}(\boldsymbol{x}, \boldsymbol{w}) := \max_{\boldsymbol{y} \in \mathbb{R}^n} \langle \mathbf{y}, \boldsymbol{x} \rangle - f^*_\varphi(\mathbf{y}, \boldsymbol{w}) - R(\mathbf{y}) = \min_{\mathbf{u} \in \mathbb{R}^n} R^*(\boldsymbol{x} - \mathbf{u}) + f_\varphi(\mathbf{u}, \boldsymbol{w})$$

and

$$\mathbf{y}_{\varphi,R}(\boldsymbol{x}, \boldsymbol{w}) := \mathbf{y}^* = \nabla R^*(\boldsymbol{x} - \mathbf{u}^*) = \nabla_1 f_{\varphi,R}(\boldsymbol{x}, \boldsymbol{w}).$$

The relaxed top-k operators are then obtained using $\varphi(x) = \frac{1}{2}x^2$ to obtain a regularized top-k in magnitude and $\varphi(x) = x$ to obtain a regularized top-k mask. We illustrate our operators in Figure 1.14.

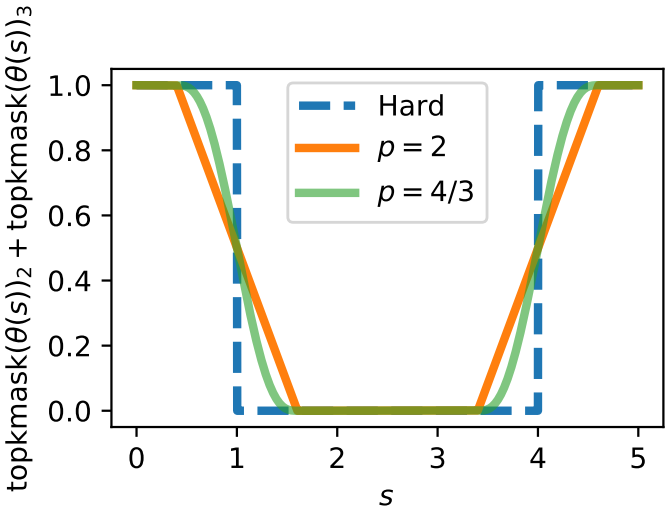

Figure 1.14: **Illustration of our differentiable and sparse top-k mask.** For $k = 2$, we consider $\boldsymbol{\theta}(s) = (3, 1, -1 + s, s) \in \mathbb{R}^4$ and plot $\text{topkmask}(\boldsymbol{\theta}(s))_2 + \text{topkmask}(\boldsymbol{\theta}(s))_3$ as a function of s. We compare the hard version (no regularization) with our proposed operator using p-norm regularization: $p = 2$ leads to differentiable a.e. operator; $p = 4/3$, leads to a differentiable operator. Both operators are **sparse**: they are exactly 0 for some values of s.

Computation. We show in Propositions 7.5 and 7.6 that the computations of our differentiable top-k operators is reduced to solving an isotonic regression problem. We leverage efficient algorithms like Pool Adjacent Violators (PAV) algorithm (Best et al., 2000), which operates in $O(n \log n)$ time, to compute the top-k operator. In cases where GPU/TPU acceleration is required, we propose using Dykstra's algorithm (Boyle and Dykstra, 1986; Combettes and Pesquet, 2011) (see Algorithm 7.7). Our implementation in JAX supports efficient gradient calculations without the need for backpropagation through unrolled iterations of the computation of the relaxed top-k operator.

Experimental results As mentioned, a key feature of our differentiable top-k operator is its ability to maintain sparsity, which is critical in applications such as sparse mixture of experts (MoEs). We experimentally show that using our differentiable top-k increases the accuracy of a sparse of mixture model on the JFT-300M dataset (Sun et al., 2017), a dataset with more than 305 million images. Our model has 32 experts, each assigned to $k = 28$ tokens selected among $n = 400$ at each MoE layer. We compare the validation accuracy when using the baseline (hard

top-k) with our relaxed operator in Figure 7.7. We use $p = 2$ and Dykstra's projection algorithm, as we found it was the fastest method on TPU.

Software. Our code is available at https://github.com/google-research/google-research/tree/master/sparse_soft_topk.

Part I

Mathematical foundations for the connection between Residual Neural Networks and Neural Ordinary Differential Equations

2

Do Residual Neural Networks discretize Neural Ordinary Differential Equations?

Neural Ordinary Differential Equations (Neural ODEs) are the continuous analog of Residual Neural Networks (ResNets). We investigate whether the discrete dynamics defined by a ResNet are close to the continuous one of a Neural ODE. We first quantify the distance between the ResNet's hidden state trajectory and the solution of its corresponding Neural ODE. Our bound is tight and, on the negative side, does not go to 0 with depth N if the residual functions are not smooth with depth. On the positive side, we show that this smoothness is preserved by gradient descent for a ResNet with linear residual functions and small enough initial loss. It ensures an implicit regularization towards a limit Neural ODE at rate $\frac{1}{N}$, uniformly with depth and optimization time. As a byproduct of our analysis, we consider the use of a memory-free discrete adjoint method to train a ResNet by recovering the activations on the fly through a backward pass of the network, and show that this method theoretically succeeds at large depth if the residual functions are Lipschitz with the input. We then show that Heun's method, a second order ODE integration scheme, allows for better gradient estimation with the adjoint method when the residual functions are smooth with depth. We experimentally validate that our adjoint method succeeds at large depth, and that Heun's method needs fewer layers to succeed. We finally use the adjoint method successfully for fine-tuning very deep ResNets without memory consumption in the residual layers.

Contents

2.1 Introduction

Problem setup. Residual Neural Networks (ResNets) (He et al., 2016a,b) keep on outperforming state of the art in computer vision (Wightman et al., 2021; Bello et al., 2021), and more generally skip connections are widely used in a various range of applications (Vaswani et al., 2017; Dosovitskiy et al., 2020). A ResNet of depth N iterates, starting from $x_0 \in \mathbb{R}^d$, $x_{n+1} = x_n + f(x_n, \theta_n^N)$ and outputs a final value $x_N \in \mathbb{R}^d$ where f is a neural network called residual function. In this work, we consider a simple modification of this forward rule by letting explicitly the residual mapping depend on the depth of the network:

$$x_{n+1} = x_n + \frac{1}{N} f(x_n, \theta_n^N). \tag{2.1}$$

On the other hand, a Neural ODE (Chen et al., 2018) uses a neural network $\varphi_\Theta(x, s)$, that takes time s into account, to parameterise a vector field (Kidger, 2022) in a differential equation, as follows,

$$\frac{\mathrm{d}x}{\mathrm{d}s} = \varphi_\Theta(x(s), s) \quad \text{with} \quad x(0) = x_0, \tag{2.2}$$

and outputs a final value $x(1) \in \mathbb{R}^d$, the solution of Eq.(2.2). The Neural ODE framework enables learning without storing activations (the x_n's) using the adjoint state method, hence significantly reducing the memory usage for backpropagation that can be a bottleneck during training (Wang et al., 2018; Peng et al., 2017; Zhu et al., 2017; Gomez et al., 2017). Neural ODEs also provide a theoretical framework to study deep learning models from the continuous viewpoint, using the arsenal of ODE theory (Teh et al., 2019; Li et al., 2019; Teshima et al., 2020). Importantly, they can also be seen as the continuous analog of ResNets. Indeed, consider for N an integer, the Euler scheme for solving Eq. (2.2) with time step $\frac{1}{N}$ starting from x_0 and iterating $x_{n+1} = x_n + \frac{1}{N}\varphi_\Theta(x_n, \frac{n}{N})$. Under mild assumptions on φ_Θ, this scheme is known to converge to the true solution of Eq. (2.2) as N goes to $+\infty$. Also, if $\Theta = (\theta_n^N)_{i\in[N-1]}$ and $\varphi_\Theta(., \frac{n}{N}) = f(., \theta_n^N)$, then the ResNet equation Eq. (2.1) corresponds to a Euler discretization with time step $\frac{1}{N}$ of Eq. (2.2). However, for a given ResNet with fixed depth N and weights, the activations in Eq. (2.1) can be far from the solution of Eq. (2.2). This is illustrated in Figure 2.1 where we show that a deep ResNet can easily break the topology of the input space, which is impossible for a Neural ODE. In this chapter, we study the link between ResNets and Neural ODEs. We make the following contributions:

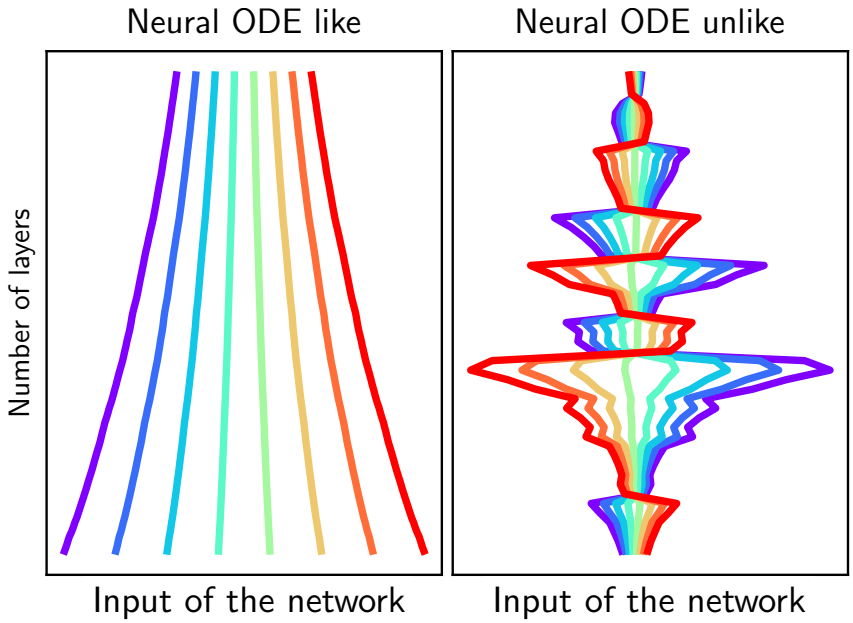

Figure 2.1: **Trajectory of ResNets with 300 layers.** Left: we learn $x \to \frac{x}{2}$, trajectories are smooth and do not intersect. Right: we learn $x \to \frac{-x}{2}$, trajectories are not smooth and intersect.

- In Section 2.3, we propose a framework to define a set of associated Neural ODEs for a given ResNet. We control the error between the discrete and the continuous trajectory. We show that without additional assumptions on the smoothness with depth of the residual functions, this error does not go to 0 as $N \to \infty$ (Prop. 2.2). However, we show that under some assumptions on the weight initialization, the trained parameters of a deep linear ResNet uniformly (with respect to both depth and training time) approach a Lipschitz function as the depth N of the network goes to infinity, at speed $\frac{1}{N}$ (Prop. 2.4 and Th. 2.5). This result highlights an implicit regularization towards a limit Neural ODE.

- In Section 2.4, we investigate a simple technique to train ResNets without storing activations. Inspired by the adjoint method, we propose to recover the approximated activations during the backward pass by using a reverse-time Euler scheme. We control the error for recovering the activations and gradients with this method. We show that if the residuals of the ResNet are bounded and Lipschitz continuous, with constants independent of N, then this error scales in $O(\frac{1}{N})$ (Prop. 2.9). Hence, the adjoint method needs a large number of layers to lead to correct gradients (Prop. 2.10). We then consider a smoothness-dependent reconstruction with Heun's method to bound the error between the true and approximated gradient by a term that depends on $\frac{1}{N}$ times the smoothness in depth of the residual functions, hence guaranteeing a better approximation when successive weights are close one to another (Prop. 2.11 and 2.12).

- In Section 2.5, on the experimental side, we show that the adjoint method fails when training a ResNet 101 on ImageNet. Nevertheless, we empirically show that very deep ResNets pretrained with tied weights (constant weights: $\theta_n^N = \theta \ \forall n$) can be refined -using our adjoint method- on CIFAR-10 and ImageNet by untying their weights, leading to a better test accuracy. Last, but not least, we show using a ResNet architecture with heavy downsampling in the first layer that our adjoint method succeeds at large depth and that Heun's method leads to a better behaved training, hence confirming our theoretical results.

2.2 Background and related work

Neural ODEs. Neural ODEs are a class of implicit deep learning models defined by an ODE where a neural network parameterises the vector field (Weinan, 2017b; Chen et al., 2018; Teh et al., 2019; Sun et al., 2018; Weinan et al., 2019; Lu et al., 2018; Ruthotto and Haber, 2019; Kidger, 2022). Given an input x_0, the output of the model is the solution of the ODE (2.2) at time 1. From a theoretical viewpoint, the expression capabilities of Neural ODEs have been investigated in (Cuchiero et al., 2020; Teshima et al., 2020; Li et al., 2019) and the Neural ODE framework has been used to better understand the dynamics of more general architectures that include residual connections such as Transformers (Sander et al., 2022a; Lu et al., 2019). Experimentaly, Neural ODEs have been successful in a various range of applications, among which physical modelling (Greydanus et al., 2019; Cranmer et al., 2019) and generative modeling (Chen et al., 2018; Grathwohl et al., 2018). However, there are many areas where Neural ODEs have failed to replace ResNets, for instance for building computer vision classification models. Neural ODEs fail to compete with ResNets on ImageNet, and to the best of our knowledge, previous works using Neural ODEs on ImageNet consider weight-tied architectures and only achieves the same accuracy as a ResNet18 (Zhuang et al., 2021). It has also been shown that Neural ODEs sometimes do not admit a continuous interpretation at all (Ott et al., 2021).

Implicit Regularization of ResNets towards ODEs. Recent works have studied the link between ResNets and Neural ODEs. In (Cohen et al., 2021), the authors carry experiments to better understand the scaling behavior of weights in ResNets as a function of the depth. They show that under the assumption that there exists a scaling limit $\theta(s) = N^\beta \lim \theta^N_{\lfloor Ns \rfloor}$ for the weights of the ResNets (with $0 < \beta < 1$) and if the scale of the ResNet is $\frac{1}{N^\alpha}$ with $0 < \alpha < 1$ and $\alpha + \beta = 1$, then the hidden state of the ResNet converges to a solution of a linear ODE. In this chapter, we are interested in the case where $\alpha = 1$, which seems more natural since it is the scaling that appears in Euler's method with step $\frac{1}{N}$. In addition, we do not assume the existence of a scaling limit $\theta(s) = \lim \theta^N_{\lfloor Ns \rfloor}$. In subsection 2.3.2, we demonstrate the existence of this scaling limit in the linear setting, under some assumptions. The recent work (Cont et al., 2022) shows results regarding linear convergence of gradient descent in ResNets and prove the existence of an $\frac{1}{2}$-Hölder continuous scaling limit as $N \to \infty$ with a scaling factor for the residuals in $\frac{1}{\sqrt{N}}$ which is different from ours ($\frac{1}{N}$). In contrast, we show that our limit function is Lipschitz continuous, which is a stronger regularity. This is to be linked with the recent work of Marion et al. (2022), where the authors show that whereas the $\frac{1}{\sqrt{N}}$ scaling corresponds to the proper one for standard i.i.d. initializations, $\frac{1}{N}$ is the proper scaling for smooth initialization to obtain non-trivial behaviour (other choices lead to explosion or to identity Marion et al. (2022)). More generally, recent works have proved the convergence of gradient descent training of ResNet when the initial loss is small enough. This include ResNet with finite width but arbitrary large depth (Du et al., 2019; Liu et al., 2020) and ResNet with both infinite width and depth (Lu et al., 2020; Barboni et al., 2021). These convergence proofs leverage an implicit bias toward weights with small amplitudes. They however leave open the question of convergence of individual weights as depth increases, which we tackle in this work in the linear case. This requires showing an extra bias toward weights with small variations across depth.

Memory bottleneck in ResNets. Training deep learning models involve graphics processing units (GPUs) where memory is a practical bottleneck (Wang et al., 2018; Peng et al., 2017; Zhu et al., 2017). Indeed, backpropagation requires to store activations at each layer during the forward pass. Since samples are processed using mini batches, this storage can be important. For instance, with batches of size 128, the memory needed to compute gradients for a ResNet 152 on

ImageNet is about 22 GiB. Note that the memory needed to store the parameters of the model is only 220 MiB, which is negligible compared to the memory needed to store the activations. Thus, designing deep invertible architectures where one can recover the activations on the fly during the backpropagation iterations has been an active field in recent years Gomez et al. (2017); Sander et al. (2021); Jacobsen et al. (2018). In this work, we propose to approximate activations using a reverse-time Euler scheme, as we detail in the next subsection.

Adjoint Method. Consider a loss function $L(x_N)$ for the ResNet (2.1). The backpropagation equations (Baydin et al., 2018) are

$$\nabla_{\theta_{n-1}^N} L = \frac{1}{N}[\partial_\theta f(x_{n-1}, \theta_{n-1}^N)]^\top \nabla_{x_n} L, \quad \nabla_{x_{n-1}} L = [I + \frac{1}{N}\partial_x f(x_{n-1}, \theta_{n-1}^N)]^\top \nabla_{x_n} L. \tag{2.3}$$

Now, consider a loss function $L(x(1))$ for the Neural ODE (2.2). The adjoint state method (Pontryagin, 1987; Chen et al., 2018) gives

$$\nabla_{\Theta(s)} L = \partial_\Theta[\varphi_\Theta(x(s), s)]^\top \nabla_{x(s)} L \mathrm{d}s, \quad -\dot{\nabla}_{x(s)} L(s) = [\partial_x \varphi_\Theta(x(s), s)]^\top \nabla_{x(s)} L. \tag{2.4}$$

Note that if $\Theta = (\theta_n^N)_{n\in[N-1]}$ and $\varphi_\Theta(., \frac{n}{N}) = f(., \theta_n^N)$, then Eq. (2.3) corresponds to a Euler discretization with time step $\frac{1}{N}$ of Eq. (2.4). The key advantage of using Eq. (2.4) is that one can recover $x(s)$ on the fly by solving the Neural ODE (2.2) backward in time starting from $x(1)$. This strategy avoids storing the forward trajectory $(x(s))_{s\in[0,1]}$ and leads to a $O(1)$ memory footprint (Chen et al., 2018). In this work, we propose to use a discrete adjoint method by using a reverse-time Euler scheme for approximately recovering the activations in a ResNet (Section 2.4). Contrarily to other models such as RevNets (Gomez et al., 2017) (architecture change) or Momentum ResNets (Sander et al., 2021) (forward rule modification) which rely on an exactly invertible forward rule, the proposed method requires no change at all in the network, but gives approximate gradients.

Notations. For $k \in \mathbb{N}$, $\mathcal{C}^k$ is the set of functions $f : \mathbb{R}^d \to \mathbb{R}^d$ k times differentiable with continuous k^{th} differential. If $f \in \mathcal{C}^1$, $\partial_x f(x)[y]$ is the differential of f at x evaluated in y. For $K \subset \mathbb{R}^d$ compact, $\|.\|$ a norm and f a continuous function on $\mathbb{R}^d$, we denote $\|f\|_\infty^K = \sup_{x\in K} \|f(x)\|$.

2.3 ResNets as discretization of Neural ODEs

In this section we first show that without further assumptions, the distance between the discrete trajectory and the solution of associated ODEs can be constant with respect to the depth of the network if the residual functions lack smoothness with depth. We then present a positive result by studying the linear case where we show that, under some hypothesis (small loss initialization and initial smoothness with depth), the ResNet converges to a Neural ODE as the number of layers goes to infinity. We show that this convergence is uniform with depth and optimization time.

2.3.1 Distance to an ODE

We first define associated Neural ODEs for a given ResNet.

Definition 2.1. *We say that a neural network $\varphi_\Theta : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ smoothly interpolates the ResNet Eq. (2.1). if φ_Θ is smooth and $\forall n \in \{0, ..., N-1\}$, $\varphi_\Theta(., \frac{n}{N}) = f(., \theta_n^N)$.*

Note that we omit the dependency of Θ in N to simplify notations. For example, for a given ResNet, there are two natural ways to interpolate it with a Neural ODE, either by interpolating the *residuals*, or by interpolating the *weights*. Indeed, one can interpolate the residuals with $\varphi_\Theta(\cdot, s) = (n+1-Ns)f(., \theta_n^N) + (Ns-n)f(\cdot, \theta_{n+1}^N)$ when $s \in [\frac{n}{N}, \frac{n+1}{N}]$, or interpolate the weights with $\varphi_\Theta(\cdot, s) = f(\cdot, (n+1-Ns)\theta_n^N + (Ns-n)\theta_{n+1}^N)$ for $s \in [\frac{n}{N}, \frac{n+1}{N}]$. If $\theta_n^N = \theta^N$ does not depend on n, then both interpolations are identical and one can simply consider $\varphi_\Theta(x, s) = f(x, \theta^N)$, $\forall (x, s)$.

We now consider any smooth interpolation φ_Θ for the ResNet (2.1) and a Euler scheme for the Neural ODE (2.2) with time step $\frac{1}{N}$.

Proposition 2.2 (Approximation error)**.** *We suppose that φ_Θ is $\mathcal{C}^1$, and L-Lipschitz with respect to x, uniformly in s. Note that this implies that the solution of Eq. (2.2) is well defined, unique, $\mathcal{C}^2$, and that the trajectory is included in some compact $K \subset \mathbb{R}^d$. Denote $C_N \coloneqq \|\partial_s \varphi_\Theta + \partial_x \varphi_\Theta[\varphi_\Theta]\|_\infty^{K \times [0,1]}$. Then one has for all n: $\|x_n - x(\frac{n}{N})\| \leq \frac{e^L - 1}{2NL} C_N$ if $L > 0$ and $\|x_n - x(\frac{n}{N})\| \leq \frac{C_N}{2N}$ if $L = 0$.*

For a full proof, see appendix 2.A.1. Note that this result extends Theorem 3.2 from (Zhuang et al., 2020) to the non-autonomous case: our bound depends on $\partial_s \varphi_\Theta$. Finally, our bound is tight. Indeed, for $\varphi_\Theta(x, s) = as + b$ for $a, b \in \mathbb{R}^d$, we get $\partial_s \varphi_\Theta + \partial_x \varphi_\Theta[\varphi_\Theta] = a$, $L = 0$ and $\|x(1) - x_N\| = \frac{\|a\|}{2N}$.

Implication. The tightness of our bound shows that closeness to the ODE solution is not guaranteed, because we do not know whether $C_N/N \to 0$. Indeed, consider first the residual interpolation $\varphi_\Theta(x, s) = \big((n+1-Ns)f(x, \theta_n^N) + (Ns-n)f(x, \theta_{n+1}^N)\big)\mathbb{1}_{s \in [\frac{n}{N}, \frac{n+1}{N}]}$ and the simple case where $\partial_x \varphi_\Theta[\varphi_\Theta] = 0$. We get $\partial_s \varphi_\Theta(x, s) = N(f(x, \theta_{n+1}^N) - f(x, \theta_n^N))\mathbb{1}_{s \in [\frac{n}{N}, \frac{n+1}{N}]}$, which corresponds to the discrete derivative. It means that although there is a $\frac{1}{N}$ factor in our bound, the time derivative term – without further regularity with depth of the weights, which is at the heart of subection 2.3.2 – usually scales with N: $\frac{C_N}{N} = O(1)$. As a first example, consider the simple case where $f(x, \theta_n^N) = n$. This gives $x_N = x_0 + \frac{(N-1)}{2}$ while the integration of the Neural ODE (2.2) leads to $x(1) = x_0 + \frac{N}{2}$ because $\varphi_\Theta(x, s) = Ns$, so the $\|x_N - x(1)\| = \frac{1}{2}$ is not small. Intuitively, this shows that weights cannot scale with depth when using the residual interpolation. Now, consider the weight interpolation, $\theta_n^N = (-1)^n$ and suppose f is written as $f(x, \theta) = \theta^2$. This gives $\varphi_\Theta(., s) = (2Ns - (2n+1))^2$ when $s \in [\frac{n}{N}, \frac{n+1}{N}]$. Integrating, we get $x(1) = \frac{1}{3}$ while the output of the ResNet is $x_N = 1$. Hence $\|x_N - x(1)\| = \frac{2}{3}$ is also not small, even though the weights are bounded. Thus, one needs additional regularity assumptions on the weights of the ResNet to obtain a Neural ODE in the large depth limit. Intuitively if the weights are initialized close from one another and they are updated using gradient descent, they should stay close from one another during training, since the gradients in two consecutive layers will be similar, as highlighted in Eq. (2.3). Indeed, we see that if x_n and x_{n+1} are close, then $\nabla_{x_n} L$ and $\nabla_{x_{n+1}} L$ are close, and then if θ_n^N and θ_{n+1}^N are close, $\nabla_{\theta_n^N} L$ and $\nabla_{\theta_{n+1}^N} L$ are also close. In subsection 2.3.2, we formalize this intuition and present a positive result for ResNets with linear residual functions. More precisely, we show that with proper initialization, the difference between two successive parameters is in $\frac{1}{N}$ during the entire training. Furthermore, we show that the weights of the network converge to a smooth function, hence defining a limit Neural ODE.

2.3.2 Linear Case

As a further step towards a theoretical understanding of the connections between ResNets and Neural ODEs we investigate the linear setting, where the residual functions are written

$f(x, \theta) = \theta x$ for any $\theta \in \mathbb{R}^{d\times d}$. It corresponds to a deep matrix factorization problem (Zou et al., 2020a; Bartlett et al., 2018; Arora et al., 2019, 2018). As opposed to these previous works, we study the infinite depth limit of these linear ResNets with a focus on the learned weights. We show that, if the weights are initialized close one to another, then at any training time, the weights stay close one to another (Prop. 2.4) and importantly, they converge to a smooth function of the continuous depth s as $N \to \infty$ (Th. 2.5). All the proofs are available in appendix 2.A.

Setting. Given a training set $(x_k, y_k)_{k\in[n]}$ in $\mathbb{R}^d$, we solve the regression problem of mapping x_k to y_k with a linear ResNet, *i.e.* $f(x, \theta) = \theta x$, of depth N and parameters $(\theta_1^N, \dots, \theta_N^N)$. The ResNet therefore maps x_k to $\Pi^N x_k$ where $\Pi^N := \prod_{n=1}^N (I_d + \frac{\theta_n^N}{N}) = (I_d + \frac{\theta_N^N}{N}) \cdots (I_d + \frac{\theta_1^N}{N})$. It is trained by minimizing the average errors $\|\Pi^N x_k - y_k\|_2^2$, which is equivalent to the deep matrix factorization problem:

$$\underset{(\theta_n^N)_{i\in[N-1]}}{\operatorname{argmin}} \quad L(\theta_1^N, \dots, \theta_N^N) := \|\Pi^N - B\|_\Sigma^2, \tag{2.5}$$

where $\|A\|_\Sigma^2 = \operatorname{Tr}(A\Sigma A^T)$, Σ is the empirical covariance matrix of the data: $\Sigma := \frac{1}{n}\sum_{k=1}^n x_k x_k^\top$, and $B := \frac{1}{n}\sum_{k=1}^n y_k x_k^\top \Sigma^{-1}$. As is standard, we suppose that Σ is non degenerated. We denote by $M > 0$ (resp. $m > 0$) its largest (resp. smallest) eigenvalue.

Gradient. We denote $\Pi_{:n}^N := (I_d + \frac{\theta_N^N}{N}) \cdots (I_d + \frac{\theta_{n+1}^N}{N})$ and $\Pi_{n:}^N := (I_d + \frac{\theta_{n-1}^N}{N}) \cdots (I_d + \frac{\theta_1^N}{N})$ and write the gradient $\nabla_n^N(t) = \nabla_{\theta_n^N} L(\theta_1^N(t), ..\theta_n^N(t), .., \theta_N^N(t))$. The chain rule gives $N\nabla_n^N = \Pi_{:n}^{N\top}(\Pi^N - B)\Sigma\Pi_{n:}^{N\top}$. Intuitively, as N goes to $+\infty$, the products Π^N, $\Pi_{:n}^N$ and $\Pi_{n:}^N$ should converge to some limit, hence we see that $N\nabla_n^N$ scale as 1. Therefore, we train θ_n^N by the rescaled gradient flow $\frac{d\theta_n^N}{dt}(t) = -N\nabla_n^N(t)$ to minimize L and denote $\ell^N(t) = L(\theta_1^N(t), \dots, \theta_N^N(t))$.

Two continuous variables involved. Our results involve two continuous variables: $s \in [0, 1]$ is the depth of the limit network and corresponds to the time variable in the Neural ODE, whereas $t \in \mathbb{R}_+$ is the gradient flow time variable. As is standard in the analysis of convergence of gradient descent for linear networks, we consider the following assumption:

Assumption 2.3. *Suppose that at initialisation one has* $\sqrt{\ell^N(0)} < \frac{m}{4\sqrt{2M}e^3}$ *and* $\|\theta_n^N(0)\| \le \frac{1}{4}$.

Assumption 2.3 is the classical assumption in the literature (Zou et al., 2020a; Barboni et al., 2021) to prove linear convergence of our loss and that the $\theta_n^N(t)$'s stay bounded with t. Note that this bounded norm assumption implies that $\frac{1}{N}\theta_n(0) = O(\frac{1}{N})$. This is in contrast with classical initialization scales in the *feedforward* case where the initialization only depends on width He et al. (2015a). However this initialization scale is coherent with those of *ResNets* for which the scale has to depend on depth (Yang and Schoenholz, 2017; Marion et al., 2022). In addition, the experimental findings in Cohen et al. (2021) suggest that the weights in ResNets scale in $\frac{1}{N^\beta}$ with $\beta > 0$.

We now prove an implicit regularization result showing that if at initialization, in addition to assumption 2.3, the weights are close from one another ($O(\frac{1}{N})$), they will stay at distance $O(\frac{1}{N})$: the discrete derivative stay in $O(\frac{1}{N})$, which is a central result to consider the infinite depth limit in our Th. 2.5.

Proposition 2.4 (Smoothness in depth of the weights)**.** *Suppose assumption 2.3. Suppose that there exists* $C_0 > 0$ *independent of* n *and* N *such that* $\|\theta_{n+1}^N(0) - \theta_n^N(0)\| \le \frac{C_0}{N}$. *Then,* $\forall t \in \mathbb{R}_+$, $\|\theta_n^N(t)\| < \frac{1}{2}$, *and* $\theta_n^N(t)$ *admits a limit* ψ_n^N *as* $t \to +\infty$. *Moreover, there exists* $C > 0$ *such that* $\forall t \in \mathbb{R}_+$, $\|\theta_{n+1}^N(t) - \theta_n^N(t)\| \le \frac{C}{N}$.

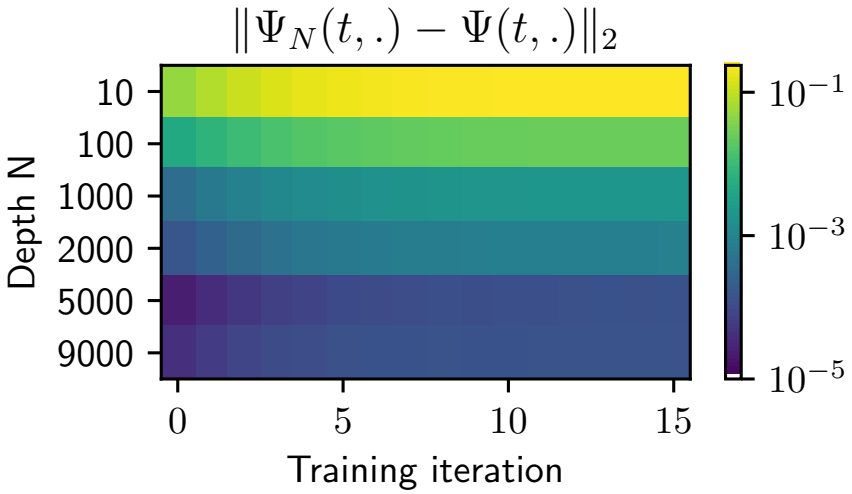

Figure 2.2: L_2 norm $\|\Psi_N(t,.) - \Psi(t,.)\|_2$ (w.r.t depth s) for different training iterations t (horizontal axis) and different depth N (vertical axis). As predicted by Th. 2.5, this distance goes to 0 as $N \to +\infty$.

For a full proof, see appendix 2.A.2. The inequality $\|\theta^N_{n+1}(t) - \theta^N_n(t)\| \leq \frac{C}{N}$ corresponds to a discrete Lipschitz property in depth. Indeed, for $s \in [0,1]$ and $t \in \mathbb{R}_+$, let $\psi_N(s,t) = \theta^N_{\lfloor Ns \rfloor}(t)$. Then our result gives $\|\psi_N(\frac{n+1}{N}, t) - \psi_N(\frac{n}{N}, t)\| \leq \frac{C}{N}$ which implies that $\|\psi_N(s_1,t) - \psi_N(s_2,t)\| \leq C|s_1 - s_2| + \frac{C}{N}$. We now turn to the infinite depth limit $N \to \infty$. Th. 2.5 shows that there exists a limit function ψ such that ψ_N converges uniformly to ψ in depth s and optimization time t. Furthermore, this limit is Lipschitz continuous in (s,t). In addition, we show that the ResNet Π^N converges to the limit Neural ODE defined by ψ that is preserved along the optimization flow, exhibiting an implicit regularization property of deep linear ResNets towards Neural ODEs.

Theorem 2.5 (Existence of a limit map)**.** *Suppose assumption 2.3, $\|\theta^N_{n+1}(0) - \theta^N_n(0)\| \leq \frac{C_0}{N}$ for some $C_0 > 0$ and that there exists a function $\psi_{\text{init}} : [0,1] \to \mathbb{R}^{d\times d}$ such that $\psi_N(s,0) \to \psi_{\text{init}}(s)$ in $\|.\|_\infty$ uniformly in s as $N \to \infty$, at speed $\frac{1}{N}$. Then the sequence $(\psi_N)_{N\in\mathbb{N}}$ uniformly converges (in $\|.\|_\infty$ w.r.t (s,t)) to a limit ψ Lipschitz continuous in (s,t) and $\|\psi - \psi_N\|_\infty = O(\frac{1}{N})$. Furthermore, Π^N uniformly converges as $N \to \infty$ to the mapping $x_0 \to x_1$ where x_1 is the solution at time 1 of the Neural ODE $\frac{\mathrm{d}x}{\mathrm{d}s}(s) = \psi(s,t)x(s)$ with initial condition x_0.*

We illustrate Th. 2.5 in Figure 2.2. The assumption on the existence of ψ_{init} ensures a convergence at speed $\frac{1}{N}$ to a Neural ODE at optimization time 0. Note that for instance, the constant initialization $\theta^N_n(0) = \theta_0 \in \mathbb{R}^{d\times d}$ satisfies this hypothesis. In order to prove Th. 2.5, for which a full proof is presented in appendix 2.A.5, we first present a useful lemma: the weights of the network have at least one accumulation point.

Lemma 2.6 (Existence of limit functions)**.** *For $s \in [0,1]$ and $t \in \mathbb{R}_+$, let $\psi_N(s,t) = \theta^N_{\lfloor Ns \rfloor}(t)$. Under the assumptions of Prop. 2.4, there exists a subsequence $\psi_{\sigma(N)}$ and $\psi_\sigma : [0,1] \times \mathbb{R}_+ \to \mathbb{R}^{d\times d}$ Lipschitz continuous with respect to both parameters s and t such that $\psi_{\sigma(N)} \to \psi_\sigma$ uniformly (in $\|.\|_\infty$ w.r.t (s,t)).*

Lemma 2.6 is proved in appendix 2.A.3, and gives us the existence of a Lipschitz continuous accumulation point, but not the uniqueness nor the convergence speed. For the uniqueness, we show in appendix 2.A.5 that, under the assumptions of Th. 2.5, one has that any accumulation point of ψ_σ satisfies the limit Neural ODE

$$\partial_t \psi_\sigma(.,t) = F(\psi_\sigma(.,t)), \quad \psi_\sigma(.,0) = \psi_{\text{init}}(.,0),$$

and show that F satisfies the hypothesis of the Picard–Lindelöf theorem, hence showing the uniqueness of ψ. We finally show that, as intuitively expected, trajectories of the weights of our linear ResNets of depth N and $2N$ remain close one to each other. This gives the convergence speed in Th. 2.5. See appendix 2.A.4 for a proof.

Lemma 2.7 (Closeness of trajectories)**.** *Suppose asumption 2.3,* $\|\theta_{n+1}^N(0) - \theta_n^N(0)\| \leq \frac{C_0}{N}$ *for some* $C_0 > 0$ *and that* $\|\theta_n^N(0) - \theta_{2n}^{2N}(0)\| = O(\frac{1}{N})$*. Then* $\forall t \in \mathbb{R}_+$*,* $\|\theta_n^N(t) - \theta_{2n}^{2N}(t)\| = O(\frac{1}{N})$*.*

2.4 Adjoint Method in Residual Networks

In this section, we focus on a particularly useful feature of Neural ODEs and its applicability to ResNets: their memory free backpropagation thanks to the adjoint method. We consider a ResNet (2.1) and try to invert it using reverse mode Euler discretization of the Neural ODE (2.2) when φ_Θ is any smooth interpolation of the ResNet. This corresponds to defining $\tilde{x}_N = x_N$ and iterate for $n \in \{N-1, \dots 0\}$:

$$\tilde{x}_n = \tilde{x}_{n+1} - \frac{1}{N} f(\tilde{x}_{n+1}, \theta_n^N). \tag{2.6}$$

We then use the approximated activations $(\tilde{x}_n)_{n\in[N]}$ as a proxy for the true activations $(x_n)_{n\in[N]}$ to compute gradients without storing the activations:

$$\tilde{\nabla}_{\theta_{n-1}^N} L = \frac{1}{N}[\partial_\theta f(\tilde{x}_{n-1}, \theta_{n-1}^N)]^\top \nabla_{\tilde{x}_n} L, \quad \nabla_{\tilde{x}_{n-1}} L = [I + \frac{1}{N}\partial_x f(\tilde{x}_{n-1}, \theta_{n-1}^N)]^\top \nabla_{\tilde{x}_n} L. \tag{2.7}$$

The approximate recovery of the activations in Eq. (2.6) is implementable for *any* ResNet: there is no need for particular architecture or forward rule modification. The drawback is that the recovery is only approximate. We devote the remainder of the section to the study of the corresponding errors and to error reduction using second order Heun's method. We first show that, if $f(., \theta_n^N)$ and its derivative are bounded by a constant independent of N, then the error for reconstructing the activations in the backward scheme (2.6) is $O(\frac{1}{N})$. Proofs of the theoretical results are in appendix 2.A.

Error for reconstructing activations. We consider the following assumption:

Assumption 2.8. *There exists constants* C_f *and* L_f *such that* $\forall N \in \mathbb{N}$*,* $\forall n \in [N-1]$*,* $\|f(., \theta_n^N)\|_\infty \leq C_f$ *and* $\|\partial_x[f(., \theta_n^N)]\|_\infty \leq L_f$*.*

Then the error made by reconstructing the activations is in $O(\frac{1}{N})$.

Proposition 2.9 (Reconstruction error)**.** *With assumption 2.8, one has* $\|x_n - \tilde{x}_n\| \leq \frac{(e^{L_f}-1)C_f}{N} + O(\frac{1}{N^2})$*.*

Prop. 2.9 shows a slow convergence of the error for recovering activations. This bound does not depend on the discrete derivative $f(., \theta_{n+1}^N) - f(., \theta_n^N)$, contrarily to the errors between the ResNet activations and the trajectory of the interpolating Neural ODE in Prop 2.2. In summary, even though regularity in depth is necessary to imply closeness to a Neural ODE, it is not necessary to recover activations, and neither gradients, as we now show.

Error in gradients when using the adjoint method. We use the result obtained in Prop. 2.9 to derive a bound in $O(\frac{1}{N^2})$ on the error made for computing gradients using formulas (2.7).

Proposition 2.10 (Gradient error)**.** *Suppose assumption 2.8. Suppose in addition that $\partial_x f(.,\theta)$ admits a Lipschitz constant L_{df}, $\partial_\theta f(.,\theta)$ admits a Lipschitz constant Δ, and an upper bound Ω, all of which are independent of θ. Then one has $\|\tilde{\nabla}_{\theta_n^N} L - \nabla_{\theta_n^N} L\| = O(\frac{1}{N^2})$.*

For a proof, see appendix 2.A.7, where we give the dependency of our upper bound as a function of Δ, L_f, C_f, Ω and L_{df}.

Smoothness-dependent reconstruction with Heun's method. The bounds in Prop. 2.9 and 2.10 do not depend on the smoothness with respect to the weights of the $f(.,\theta_n^N)$. Only the magnitude of the residuals plays a role in the correct recovery of the activations and estimation of the gradient. Hence, there is no apparent benefit of having such a network behave like a Neural ODE. We now turn to Heun's method, a second order integration scheme, and show that in this case smoothness in depth of the network improves activation recovery. A HeunNet (Maleki et al., 2021) of depth N with parameters $\theta_1^N, \dots, \theta_N^N$ iterates for $n = 0, \dots, N-1$:

$$y_n = x_n + \frac{1}{N} f(x_n, \theta_n^N) \quad \text{and} \quad x_{n+1} = x_n + \frac{1}{2N}(f(x_n, \theta_n^N) + f(y_n, \theta_{n+1}^N)). \tag{2.8}$$

These forward iterations can once again be approximately reversed by doing for $n = N-1, \dots, 0$:

$$\tilde{y}_n = \tilde{x}_{n+1} - \frac{1}{N} f(\tilde{x}_{n+1}, \theta_{n+1}^N) \quad \text{and} \quad \tilde{x}_n = \tilde{x}_{n+1} - \frac{1}{2N}(f(\tilde{x}_{n+1}, \theta_{n+1}^N) + f(\tilde{y}_n, \theta_n^N)), \tag{2.9}$$

which also enables approximated backpropagation without storing activations. When discretizing an ODE, Heun's method has a better $O(\frac{1}{N^2})$ error, hence we expect a better recovery than in Prop. 2.9. Indeed, we have:

Proposition 2.11 (Reconstruction error - Heun's method)**.** *Assume assumption 2.8. Denote by L'_f the Lipschitz constant of $x \mapsto \frac{1}{2}(f(x, \theta_{n+1}^N) + f(x - \frac{1}{N} f(x, \theta_{n+1}^N), \theta_n^N))$, by L_θ the Lispchitz constant of $\theta \mapsto f(\cdot, \theta)$ and by L'_θ that of $\theta \mapsto \partial_x f(.,\theta)$. Let $C'_f = \frac{1}{4} L'_\theta L_\theta$. Finally, define $\Delta_\theta^N := \max_n \|\theta_{n+1}^N - \theta_n^N\|^2$. Using Heun's method, we have: $\|x_n - \tilde{x}_n\| \leq \frac{(e^{L'_f} - 1)C'_f}{L'_f N} \times \Delta_\theta^N + O(\frac{1}{N^2})$.*

This bound is very similar to that in proposition 2.9, with an additional factor Δ_θ^N. Hence, we see that under the condition that $\Delta_\theta^N = O(\frac{1}{N})$, the reconstruction error $\|x_n - \tilde{x}_n\|$ is in $O(\frac{1}{N^2})$. In the linear case, we have proven under some hypothesis in Prop. 2.4 that such a condition on Δ_θ^N holds during training. Consequently, the smoothness of the weights of a HeunNet in turns helps it recover the activations, while it is not true for a ResNet. This provides better guarantees on the error on gradients:

Proposition 2.12 (Gradient error - Heun's method)**.** *Suppose assumption 2.8. Suppose in addition that $\partial_x f(.,\theta)$ admits Lipschitz constant, $\partial_\theta f(.,\theta)$ admits a Lipschitz constant and an upper bound, all of which are independent of θ. Then one has $\|\tilde{\nabla}_{\theta_n^N} L - \nabla_{\theta_n^N} L\| = O(\frac{\Delta_\theta^N}{N^2} + \frac{1}{N^3})$.*

Just like with activation, we see that Heun's method allows for a better gradient estimation when the weights are smooth with depth. Equivalently, for a fixed depth, this proposition indicates that HeunNets have a better estimation of the gradient with the adjoint method than ResNets which ultimately leads to better training and overall better performances by such memory-free model.

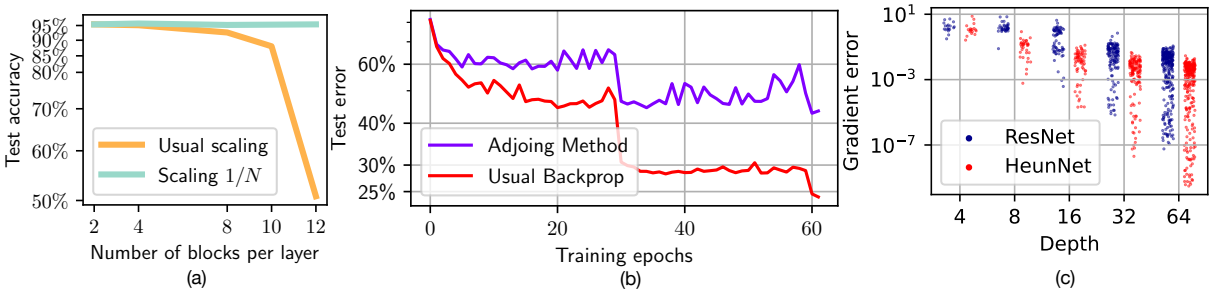

Figure 2.3: (a) **Test accuracy on CIFAR-10** as a function of the number of blocks in each layer of the ResNet. Within each layer, weights are tied (3 runs). (b) **Failure** of the adjoint method with a ResNet-101 on ImageNet (the approximated gradients are only used in the third layer of the network, that contains 23 blocks). (c) **Relative error** between the approximated gradients using adjoint method and the true gradient, whether using a ResNet or a HeunNet. Each point corresponds to one parameter.

Table 2.1: Test accuracy (ResNet-101)

	ResNet-101	Ours
CIFAR-10	$95.5 \pm 0.1\%$	$95.5 \pm 0.1\%$
ImageNet	77.8%	77.9%

2.5 Experiments

We now present experiments to investigate the applicability of the results presented in this chapter. We use Pytorch (Paszke et al., 2017) and Nvidia Tesla V100 GPUs. Our code is available on GitHub. All the experimental details are given in appendix 2.B, and we provide a recap on ResNet architectures in appendix 2.C.

2.5.1 Validation of our scaled model

The ResNet model (2.1) is different from the classical ResNet because of the $\frac{1}{N}$ term. This makes the model depth aware, and we want to study the impact of this modification on the accuracy on CIFAR and ImageNet. We first train a ResNet-101 (He et al., 2016a) on CIFAR-10 and ImageNet using the same hyper-parameters. Experimental details are in appendix 2.B and results are summarized in table 2.1, showing that the explicit addition of the step size $\frac{1}{N}$ does not affect accuracy. In strike contrast, the classical ResNet rule without the scaling $\frac{1}{N}$ makes the network behave badly at large depth, while it still works well with our scaling $\frac{1}{N}$, as shown in Figure 2.3 (a). On ImageNet, the scaling $\frac{1}{N}$ also leads to similar test accuracy in the weight tied setting: 72.5% with 4 blocks per layer, 73.2% with 8 blocks per layer and 72% with 16 blocks per layer (mean over 2 runs).

2.5.2 Adjoint method

New training strategy. Our results in Prop. 2.9 and 2.10 assume uniform bounds in N on our residual functions and their derivatives. We also formally proved in the linear setting that these assumptions hold during the whole learning process if the initial loss is small. A natural idea to start from a small loss is to consider a pretrained model. In addition, we also want our pretrained model to verify assumption 2.8 so we consider the following setup. On CIFAR (resp. ImageNet) we train a ResNet with 4 (resp. 8) blocks in each layer, where weights are tied within each layer. A first observation is that one can transfer these weights to deeper ResNets without

Table 2.2: Test accuracy (ResNet)

	Before F.T.	After F.T
CIFAR-10	95.25 ± 0.2 %	95.65 ± 0.1 %
ImageNet	73.1 %	75.1 %

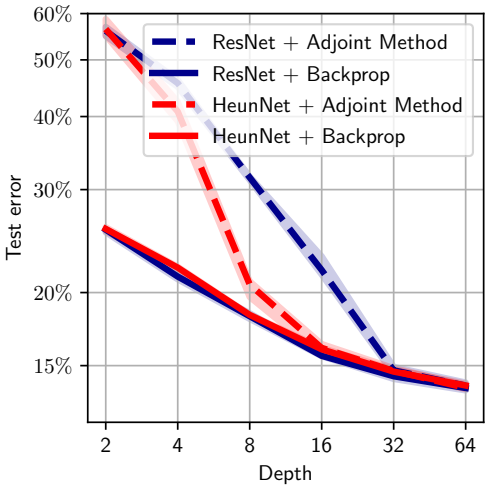

Figure 2.4: **Comparison** of the best test errors as a function of depth when using Euler or Heun's discretization method with or without the adjoint method.

significantly affecting the test accuracy of the model: it remains above 94.5% on CIFAR-10 and 72% on ImageNet. We then *untie the weights* of our models and refine them. More precisely, for CIFAR, we then transfer the weights of our model to a ResNet with 4, 4, 64 and 4 blocks within each layer and fine-tune it only by refining the third layer, using our adjoint method. We display in table 2.2 the median of the new test accuracy, over 5 runs for the initial pretraining of the model. For ImageNet, we transfer the weights to a ResNet with 100 blocks per layer and fine-tune the whole model with our adjoint method for the residual layers. Results are summarized in table 2.2. To the best of our knowledge, this is the first time a Neural-ODE like ResNet achieves a test-accuracy of 75.1% on ImageNet.

Failure in usual settings. In Prop. 2.9 we showed under assumption 2.8, that is if the residuals are bounded and Lipschitz continuous with constant independent of the depth N, then the error for computing the activations backward would scale in $\frac{1}{N}$ as well as the error for the gradients (Prop. 2.10). First, this results shows that the architecture needs to be deep enough, because it scales in $\frac{1}{N}$: for instance, we fail to train a ResNet-101 (He et al., 2016a) on the ImageNet dataset using the adjoint method on its third layer (depth 23), as shown in Figure 2.3 (b).

Success at large depth. To further investigate the applicability of the adjoint method for training deeper ResNets, we train a simple ResNet model on the CIFAR data set. First, the input is processed by a 5×5 convolution with 16 out channels, and the image is down-sampled to a size 10×10. We then apply a batch norm, a ReLU and iterate relation (2.1) where f is a pre-activation basic block (He et al., 2016b). We consider the zero residual initialisation: the last batch norm of each basic block is initialized to zero. We consider different values for the depth N and notice that in this setup, the deeper our model is, the better it performs in term of test

accuracy. We then compare the performance of our model using a ResNet (forward rule (2.1)) or a HeunNet (forward rule (2.8)). We train our networks using either the classical backpropagation or our corresponding proxys using the adjoint method (formulas (2.6) and (2.9)). We display the final test accuracy (median over 5 runs) for different values of the depth N in Figure 2.4. The true backpropagation gives the same curves for the ResNet and the HeunNet. Approximated gradients, however, lead to a large test error at small depth, but give the same performance at large depth, hence confirming our results in Prop. 2.10 and 2.12. In addition, at fixed depth, the accuracy when training a HeunNet with the adjoint method is better (or similar at depths 2, 32 and 64) than for the ResNet with the adjoint method. This is to be linked with the two different bounds in Prop. 2.10 and 2.12: for the HeunNet, smoothness with depth, which is expected at large depth, according to the theoretical results for the linear case (Prop. 2.4), implies a faster convergence to the true gradients for the HeunNet than for the ResNet. We finally validate this convergence in Figure 2.3 (c): the deeper the architecture, the better the approximation on the gradients. In addition, the HeunNet approximates the true gradient better than the ResNet.

Conclusion, limitations and future works

We propose a methodology to analyze how well a ResNet discretizes a Neural ODE. The positive results predicted by our theory in the linear case are also observed in practice with real architectures: one can successfully use the adjoint method to train ResNets (or even more effectively HeunNets) using very deep architectures on CIFAR, or fine-tune them on ImageNet, without memory cost in the residual layers. However, we also show that for large scale problems such as ImageNet classification from scratch, the adjoint method fails at usual depths.

Our work provides a theoretical guarantee for the convergence to a Neural ODE in the linear setting under a small loss initialization. A natural extension would be to study the non-linear case. In addition, the adjoint method is time consuming, and an improvement would be to propose a cheaper method than a reverse mode traversal of the architecture for approximating the activations.

In Section 2.A we give the proofs of all the propositions, lemmas and the theorem presented in this work. Section 2.B gives details for the experiments in the chapter. We also give a recap on ResNet architectures in Section 2.C.

2.A Proofs

2.A.1 Proof of Prop. 2.2

Our proof is inspired by (Demailly, 2016).

Proof. We denote $h = \frac{1}{N}$ and $s_n = nh$. We define

$$\varepsilon_n = x(s_{n+1}) - x(s_n) - h\varphi_\Theta(x(s_n), s_n).$$

We have that $\varphi_\Theta(x(s_n), s_n) = \dot{x}(s_n)$.

Taylor's formula gives

$$x(s_n + h) = x(s_n) + h\dot{x}(s_n) + R_1(h)$$

with $\|R_1(h)\| \leq \frac{1}{2}h^2\|\ddot{x}\|_\infty$. This implies that

$$\|\varepsilon_n\| \leq \frac{1}{2}h^2\|\ddot{x}\|_\infty.$$

The true error we are interested in is the global error $e_n = x(s_n) - x_n$. One has

$$e_{n+1} - e_n = x(s_{n+1}) - x(s_n) + x_n - x_{n+1} = \varepsilon_n + h(\varphi_\Theta(x(s_n), s_n) - \varphi_\Theta(x_n, s_n)).$$

Because φ_Θ is L-Lipschitz, this gives $\|e_{n+1} - e_n\| \leq \|\varepsilon_n\| + hL\|e_n\|$ and hence

$$\|e_{n+1}\| \leq (1 + hL)\|e_n\| + \|\varepsilon_n\|.$$

Because $h = \frac{1}{N}$, we have

$$\|e_{n+1}\| \leq (1 + \frac{L}{N})\|e_n\| + \frac{1}{2N^2}\|\ddot{x}\|_\infty.$$

this implies from the discrete Gronwall lemma, since $e_0 = 0$ that

$$\|e_n\| \leq \frac{e^L - 1}{2NL}\|\ddot{x}\|_\infty.$$

Note that we have $\ddot{x} = \partial_s\varphi_\Theta + \partial_x\varphi_\Theta[\varphi_\Theta]$. This gives the desired result. □

2.A.2 Proof of Prop. 2.4

Proof. Recall that we denote $\Pi^N = \prod_{n=1}^N (I_d + \frac{\theta_n^N}{N})$, $\Pi_{:n}^N = (I_d + \frac{\theta_N^N}{N}) \dots (I_d + \frac{\theta_{n+1}^N}{N})$ and $\Pi_{n:}^N = (I_d + \frac{\theta_{n-1}^N}{N}) \dots (I_d + \frac{\theta_1^N}{N})$. We denote $\nabla_n^N = \nabla_{\theta_n^N} L$. One has

$$N\nabla_n^N = \Pi_{:n}^{N\top}(\Pi^N - B)\Sigma\Pi_{n:}^{N\top}.$$

One has as in (Zou et al., 2020a) that

$$\sigma_{max}^2(\Pi_{n:}^N)\sigma_{max}^2(\Pi_{:n}^N)\|\Pi - B\|_\Sigma^2 M \geq N^2\|\nabla_n^N\|^2 \geq \sigma_{min}^2(\Pi_{n:}^N)\sigma_{min}^2(\Pi_{:n}^N)\|\Pi - B\|_\Sigma^2 m.$$

where $\sigma_{max}(A)$ (resp. $\sigma_{min}(A)$) denotes the largest (resp. smallest) singular value of A. We first show that $\forall t \in \mathbb{R}_+$, $\|\theta_n^N(t)\| < \frac{1}{2}$. Denote

$$t^* = \inf\{t \in \mathbb{R}_+, \exists n \in [N-1], \|\theta_n^N(t)\| \geq 1/2\}.$$

One has that $\forall t \in [0, t^*]$, $\sigma_{min}^2(\Pi_{:n}^N) \geq (1 - \frac{1}{2N})^{2(N-n)}$ and $\sigma_{min}^2(\Pi_{n:}^N) \geq (1 - \frac{1}{2N})^{2(n-1)}$ which implies that

$$N^2\|\nabla_n^N(t)\|^2 \geq 2(1 - \frac{1}{2N})^{2N-2}\ell^N(t)m \geq \frac{2}{e}\ell^N(t)m.$$

Similarly one has $N^2\|\nabla_n^N(t)\|^2 \leq 2e\ell^N(t)M$. To summarize, we have the PL conditions for $t \in [0, t^*]$:

$$\frac{2}{e}m\ell^N(t) \leq N^2\|\nabla_n^N(t)\|^2 \leq 2eM\ell^N(t).$$

As a consequence, one has

$$\frac{\mathrm{d}\ell^N}{\mathrm{d}t}(t) = -N\sum_{n=1}^{N}\|\nabla_n^N(t)\|^2 \leq -\frac{2}{e}m\ell^N(t)$$

and thus $\ell^N(t) \leq e^{-\frac{2}{e}mt}\ell^N(0)$.

We have $\theta_n^N(t^*) = \theta_n^N(0) + N\int_0^{t^*}\nabla_n^N$ and $\|\nabla_n^N\| \leq \frac{\sqrt{2eM}}{N}\sqrt{\ell^N}$ so that

$$\|\theta_n^N(t^*)\| \leq \|\theta_n^N(0)\| + \sqrt{2eM}\int_0^{t^*} e^{-\frac{1}{e}mt}\sqrt{\ell^N(0)}dt < \frac{1}{4} + \frac{1}{4} < 1/2.$$

This is absurd by definition of t^* and thus shows that $\forall t \in \mathbb{R}_+, \|\theta_n^N(t)\| < \frac{1}{2}$. We also see that ∇_n^N is integrable so that $\theta_n^N(t)$ admits a limit as $t \to \infty$.

We now show our main result. Note that we have the relationship $(I + \frac{\theta_{n+1}^N}{N})^\top\nabla_{n+1} = \nabla_n(I + \frac{\theta_n^N}{N})^\top$ so that

$$\nabla_{n+1}^N - \nabla_n^N = (I + \frac{\theta_{n+1}^{N\top}}{N})^{-1}(\frac{\nabla_n^N\theta_n^{N\top} - \theta_{n+1}^{N\top}\nabla_n^N}{N}).$$

Because $\|(I + A)^{-1}\| \leq 2$ if $\|A\| \leq \frac{1}{2}$ this gives $\|\nabla_{n+1}^N - \nabla_n^N\| \leq \frac{2}{N}\|\nabla_n^N\|$. Integrating we get

$$\|\theta_{n+1}^N(t) - \theta_n^N(t)\| \leq \|\theta_{n+1}^N(0) - \theta_n^N(0)\| + 2\int_0^t\|\nabla_n^N\|.$$

This gives

$$\|\theta_{n+1}^N(t) - \theta_n^N(t)\| \leq O(\frac{1}{N}) + \frac{1}{N}2\int_0^t\sqrt{2eM\ell^N(0)}e^{-\frac{1}{e}mt}dt = O(\frac{1}{N}),$$

which is the desired result.

□

2.A.3 Proof of lemma 2.6

Proof. We adapt a variant of the Ascoli–Arzelà theorem (Brezis and Brézis, 2011). We showed in Prop. 2.4 that there exists $C > 0$ that only depends on the initialization such that, $\forall t \geq 0, \forall i \in [N-1]$,

$$\|\theta_{n+1}^N(t) - \theta_n^N(t)\| \leq \frac{C}{N}.$$

This implies that

$$\|\theta_j^N(t) - \theta_i^N(t)\| \leq C\frac{|j-i|}{N}.$$

We also have that

$$\|\theta_n^N(t_1) - \theta_n^N(t_2)\| = \|N\int_{t_1}^{t_2} \nabla_n^N\| \leq C'|t_1 - t_2|$$

with $C' \geq 0$.

Its follows that $\|\psi_N(s_1,t_1) - \psi_N(s_2,t_2)\| \leq \|\psi_N(s_1,t_1) - \psi_N(s_1,t_2)\| + \|\psi_N(s_1,t_2) - \psi_N(s_2,t_2)\|$ and thus

$$(i) \quad \|\psi_N(s_1,t_1) - \psi_N(s_2,t_2)\| \leq C'|t_1 - t_2| + C|s_1 - s_2| + \frac{C}{N}.$$

We also have

$$(ii) \quad \forall N \in \mathbb{N}, \quad \|\psi_N\|_\infty \leq \frac{1}{2}.$$

These two properties are essential to prove our lemma. We proceed as follows.

1) First, we denote $((s_j,t_j))_{j\in\mathbb{N}} = (\mathbb{Q}\cap[0,1]) \times \mathbb{Q}_+$. Since we have the uniform bound (ii), we extract using a diagonal extraction procedure a subsequence $\psi_{\sigma(N)}$ such that $\forall j \in \mathbb{N}$,

$$\psi_{\sigma(N)}(s_j,t_j) \to \psi(s_j,t_j)$$

(we denote the limit $\psi(s_j,t_j)$).

2) We show the convergence $\forall s \in [0,1]$ and $t \in \mathbb{R}_+$.

Let $\varepsilon > 0$, $s \in [0,1]$ and $t \in \mathbb{R}_+$. Since $((s_j,t_j))_{j\in\mathbb{N}}$ is dense in $[0,1]\times\mathbb{R}_+$, there exists $k \in \mathbb{N}$ such that $|s_k - s| < \varepsilon$ and $|t_k - t| < \varepsilon$. Let $N, M \in \mathbb{N}$.

We have

$\|\psi_{\sigma(N)}(s,t) - \psi_{\sigma(M)}(s,t)\| \leq \|\psi_{\sigma(N)}(s,t) - \psi_{\sigma(N)}(s_k,t_k)\| + \|\psi_{\sigma(N)}(s_k,t_k) - \psi_{\sigma(M)}(s_k,t_k)\| + \|\psi_{\sigma(M)}(s_k,t_k) - \psi_{\sigma(M)}(s,t)\|$

so that

$$\|\psi_{\sigma(N)}(s,t) - \psi_{\sigma(M)}(s,t)\| \leq 2C\varepsilon + 2C'\varepsilon + \frac{C}{\sigma(N)} + \frac{C}{\sigma(M)} + \|\psi_{\sigma(N)}(s_k,t_k) - \psi_{\sigma(M)}(s_k,t_k)\|.$$

Since $(\psi_{\sigma(N)}(s_k,t_k))_{N\in\mathbb{N}}$ is a Cauchy sequence, this gives for N, M big enough that

$$\|\psi_{\sigma(N)}(s) - \psi_{\sigma(M)}(s,t)\| \leq (2(C+C')+1)\varepsilon$$

and thus $(\psi_{\sigma(N)}(s,t))$ is a Cauchy sequence in $\mathbb{R}^{d\times d}$. As such, it converges and one has

$$\psi_{\sigma(N)}(s,t) \to \psi(s,t).$$

3) Recall that one has

$$\|\psi_{\sigma(N)}(s_1,t_1) - \psi_{\sigma(N)}(s_2,t_2)\| \leq C|s_1 - s_2| + \frac{C}{\sigma(N)} + C'|t_1 - t_2|$$

so that letting $N \to \infty$ gives

$$\|\psi(s_1,t_1) - \psi(s_2,t_2)\| \leq C|s_1 - s_2| + C'|t_1 - t_2|$$

and ψ is Lipschitz continuous.

4) Let us finally show that the convergence is uniform in (s,t). Let $s \in [0,1]$, $\varepsilon > 0$ and $\delta > 0$ such that if $|s-u| < \delta$, $\forall t \in \mathbb{R}+$,

$$\|\psi_N(s,t) - \psi_N(u,t)\| \leq \varepsilon + \frac{C}{N}$$

and $\|\psi(s,t) - \psi(u,t)\| \leq \varepsilon$. There exists a finite set of $\{s_j\}_{j=1}^k$ such that

$$[0,1] \subset \cup_{j=1}^k]s_j - \frac{\delta}{2}, s_j + \frac{\delta}{2}[.$$

For our s, there exists $j \in \{1,\dots,k\}$ such that $\|s - s_j\| \leq \delta$.

There also exists $t_0 \geq 0$ such that if $t \geq t_0$,

$$\|\psi_{\sigma(N)}(s,t) - \psi_{\sigma(N)}(s,t_0)\| \leq \sqrt{2eM} \int_{t_0}^t e^{-\frac{1}{e}mz} \sqrt{\ell^N(0)} dz \leq \varepsilon.$$

We have:

$\|\psi_{\sigma(N)}(s,t_0) - \psi(s,t_0)\| \leq \|\psi_{\sigma(N)}(s,t_0) - \psi_{\sigma(N)}(s_j,t_0)\| + \|\psi_{\sigma(N)}(s_j,t_0) - \psi(s_j,t_0)\| + \|\psi(s_j,t_0) - \psi(s,t_0)\|$

and thus:

$\|\psi_{\sigma(N)}(s,t_0) - \psi(s,t_0)\| \leq 2\varepsilon + \frac{C}{\sigma(N)} + \max_{j\in\{1,\dots,k\}} \|\psi_{\sigma(N)}(s_j,t_0) - \psi(s_j,t_0)\| \leq 4\varepsilon$ for N big enough.

Finally, $\|\psi_{\sigma(N)}(s,t) - \psi(s,t)\| \leq \|\psi_{\sigma(N)}(s,t) - \psi_{\sigma(N)}(s,t_0)\| + \|\psi_{\sigma(N)}(s,t_0) - \psi(s,t_0)\| + \|\psi(s,t_0) - \psi(s,t)\| \leq 6\varepsilon$

for N big enough, independently of t and s. This concludes the proof. □

2.A.4 Proof of lemma 2.7

Proof. We group terms 2 by 2 in the product Π^{2N}. One has $(I + \frac{\theta_{2n}^{2N}}{2N})(I + \frac{\theta_{2n-1}^{2N}}{2N}) = (I + \frac{\tilde{\theta}_n^N}{N})$ with

$$\tilde{\theta}_n^N = (\frac{\theta_{2n}^{2N} + \theta_{2n-1}^{2N}}{2} + \frac{\theta_{2n}^{2N}\theta_{2n-1}^{2N}}{4N}),$$

So that $\Pi^{2N} = \tilde{\Pi}^N$ where $\tilde{\Pi}^N$ is defined as Π^N with $\tilde{\theta}_n^N$. One has by Prop. 2.4 that

$$\tilde{\theta}_n^N = \theta_{2n}^{2N} + O(\frac{1}{N}).$$

We will show that $\tilde{\theta}_n^N = \theta_n^N + O(\frac{1}{N})$.

Let $D_n^N = \|\theta_n^N - \tilde{\theta}_n^N\|$ and $D^N = \frac{1}{N}\sum_{n=1}^N D_n$. We have

$$2D_n^N \dot{D}_n^N = -N\langle \nabla_n^N - \tilde{\nabla}_n^N, \theta_n^N - \tilde{\theta}_n^N \rangle.$$

In addition, we have

$$N(\nabla_n^N - \tilde{\nabla}_n^N) = \Pi_{:n}^{N\top}(\Pi^N - B)\Sigma\Pi_{n:}^{N\top} - \tilde{\Pi}_{:n}^{N\top}(\tilde{\Pi}^N - B)\Sigma\tilde{\Pi}_{n:}^{N\top}$$

so that

$$N(\nabla_n^N - \tilde{\nabla}_n^N) = (\Pi_{:n}^N - \tilde{\Pi}_{:n}^N)^\top (\Pi^N - B)\Sigma \Pi_{n:}^{N\top} + \tilde{\Pi}_{:n}^{N\top}(\Pi^N - B)\Sigma(\Pi_{n:}^N - \tilde{\Pi}_{n:}^N)^\top + \tilde{\Pi}_{:n}^{N\top}(\Pi^N - \tilde{\Pi}^N)\Sigma \tilde{\Pi}_{n:}^{N\top}.$$

Note also that since the Jacobian of $(\theta_1, .., \theta_N) \to \Pi^N$ is

$$J_{(\theta_1,..,\theta_N)}(H_1, .., H_N) = \frac{1}{N}\sum_{n=1}^N \Pi_{:n}^N H_n \Pi_{n:}^N$$

and the θ_n^N's are such that $\|\theta_n^N\| \leq \frac{1}{2}$, there exists a constant $K > 0$ such that $\|\Pi_{:n}^N - \tilde{\Pi}_{:n}^N\| \leq KD^N$. Again because $\|\theta_n^N\| \leq \frac{1}{2}$ and $\|\tilde{\theta}_n^N\| \leq \frac{1}{2}$, this gives

$$N\|\nabla_n - \tilde{\nabla}_n\| \leq \alpha K D^N \sqrt{\ell^N} + \beta \|\Pi^N - \tilde{\Pi}^N\|$$

for some constants α, β. Finally, we have

$$\dot{D}_n^N \leq \frac{1}{2}(\alpha K D^N \sqrt{\ell^N} + \beta \|\Pi^N - \tilde{\Pi}^N\|)$$

which gives $\forall t$

$$2D_n^N(t) \leq \alpha K \int_0^t D^N \sqrt{\ell^N} + \beta \int_0^t \|\Pi^N - \tilde{\Pi}^N\| + O(\frac{1}{N}). \tag{2.10}$$

We now focus on the β term involving $\|\Pi^N - \tilde{\Pi}^N\|$. Denote $\Delta^N = \Pi^N - \tilde{\Pi}^N$. One has

$$\dot{\Delta}^N = -\frac{1}{N}(\sum_{n=1}^N \Pi_{:n}^N \Pi_{:n}^{N\top}(\Pi^N - B)\Sigma \Pi_{n:}^{N\top}\Pi_{n:}^N + \tilde{\Pi}_{:n}^N \tilde{\Pi}_{:n}^{N^\top}(\tilde{\Pi}^N - B)\Sigma \tilde{\Pi}_{n:}^{N\top}\tilde{\Pi}_{n:}^N),$$

and equivalently:

$\dot{\Delta}^N = -\frac{1}{N}(\sum_{n=1}^N [\Pi_{:n}^N \Pi_{:n}^{N\top} - \tilde{\Pi}_{:n}^N \tilde{\Pi}_{:n}^{N\top}](\Pi^N - B)\Sigma \Pi_{n:}^{N\top}\Pi_{n:}^N + \tilde{\Pi}_{:n}^N \tilde{\Pi}_{:n}^{N\top}(\Pi^N - B)\Sigma[\Pi_{n:}^{N\top}\Pi_{n:}^N - \tilde{\Pi}_{n:}^{N\top}\tilde{\Pi}_{n:}^N] + \tilde{\Pi}_{:n}^N \tilde{\Pi}_{:n}^{N\top}(\Pi^N - \tilde{\Pi}^N)\Sigma \tilde{\Pi}_{n:}^{N\top}\tilde{\Pi}_{n:}^N)$.

Note that similarly to $\|\Pi^N - \tilde{\Pi}^N\|$ there exist K' such that $\|\Pi_{:n}^N \Pi_{:n}^{N\top} - \tilde{\Pi}_{:n}^N \tilde{\Pi}_{:n}^{N\top}\| \leq K'D$ so that

$$\|\dot{\Delta}^N + \frac{1}{N}\sum_{n=1}^N \tilde{\Pi}_{:n}^N \tilde{\Pi}_{:n}^{N\top} \Delta^N \tilde{\Pi}_{n:}^{N\top}\tilde{\Pi}_{n:}^N\| \leq aK'D\sqrt{\ell^N}.$$

Let us denote by H the operator:

$$H(\Delta) = \frac{1}{N}\sum_{n=1}^N \tilde{\Pi}_{:n}^N \tilde{\Pi}_{:n}^{N\top} \Delta \Sigma \tilde{\Pi}_{n:}^{N\top}\tilde{\Pi}_{n:}^N.$$

Our (PL) conditions precisely write $-\Delta^\top H(\Delta) \leq -\lambda\|\Delta\|^2$ for some $\lambda > 0$. Let $\varphi^N = \frac{1}{2}\|\Delta^N\|^2$. One has

$$\frac{\mathrm{d}\varphi^N}{\mathrm{d}t} = \langle \Delta^N, \dot{\Delta}^N + H(\Delta^N)\rangle - \langle \Delta^N, H(\Delta^N)\rangle$$

so that

$$\frac{\mathrm{d}\varphi^N}{\mathrm{d}t} \leq (aK'D\sqrt{\ell^N})\sqrt{2\varphi^N} - 2\lambda\varphi^N.$$

Since $\|\Delta^N\| = \sqrt{2\varphi^N}$ we get

$$\frac{\mathrm{d}\|\Delta^N\|}{\mathrm{d}t} = \frac{\frac{\mathrm{d}\varphi^N}{\mathrm{d}t}}{\sqrt{2\varphi^N}}.$$

We finally have

$$\frac{\mathrm{d}\|\Delta^N\|}{\mathrm{d}t} \leq aK'D^N\sqrt{\ell^N} - \lambda\|\Delta^N\|.$$

Integrating, we get

$$\|\Delta^N(t)\| \leq -\lambda\int_0^t \|\Delta^N\| + \int_0^t aK'D^N\sqrt{l^N} + O(\frac{1}{N})$$

and then

$$\int_0^t \|\Delta^N\| \leq \frac{1}{\lambda}\int_0^t aK'D^N\sqrt{l^N} + O(\frac{1}{N}).$$

Plugging this into (2.10) leads to

$$0 \leq 2D_n^N(t) \leq \alpha K\int_0^t D^N\sqrt{\ell^N} + \frac{\beta}{\lambda}\int_0^t aK'D^N\sqrt{\ell^N} + O(\frac{1}{N}).$$

Let $n(t)$ be such that $D^N_{n(t)}(t) = \max_{i\in[1,N]} D_i^N(t)$. We have

$$0 \leq 2D^N_{n(t)}(t) \leq \mu\int_0^t D^N_{n(\tau)}(\tau)\sqrt{\ell^N(\tau)}d\tau + O(\frac{1}{N})$$

for some constant $\mu > 0$. And since $\sqrt{\ell^N}$ is integrable, we get by Gronwall's inequality that $D_n^N = O(\frac{1}{N})\ \forall n \in [1, N]$. We showed:

$$\theta^{2N}_{2n} = \theta^N_n + O(\frac{1}{N}).$$

□

2.A.5 Proof of Th. 2.5

We first prove the following lemma 2.13 before proving Th. 2.5.

Lemma 2.13. *Under the assumptions of Th. 2.5, let σ be such that $\psi_{\sigma(N)} \to \psi_\sigma$ uniformly (in $\|.\|_\infty$ w.r.t (s,t)). Then one has $\Pi^{\sigma(N)}(t) \to \Pi(t)$ uniformly (in t) where $\Pi(t)$ maps x_0 to the solution at time 1 of the Neural ODE $\frac{\mathrm{d}x}{\mathrm{d}s} = \psi_\sigma(s,t)x(s)$ with initial condition x_0.*

Proof. Consider for $x_0 \in \mathbb{R}^d$ with $\|x_0\| = 1$ the discrete scheme

$$x_{n+1} = x_n + \frac{1}{\sigma(N)}\theta_n^{\sigma(N)}(t)x_n,$$

the ODE

$$\frac{\mathrm{d}x}{\mathrm{d}s} = \psi_\sigma(s,t)x(s),$$

and the Euler scheme with time step $\frac{1}{\sigma(N)}$ for its discretization

$$y_{n+1} = y_n + \frac{1}{\sigma(N)}\psi_\sigma(\frac{n}{\sigma(N)}, t)y_n.$$

We know by Prop. 2.2, since x_0 has unit norm that

$$\|x(\frac{n}{\sigma(N)}) - y_n\| \leq \frac{e^{\frac{1}{2}} - 1}{\sigma(N)}\|\partial_s\psi_\sigma(.,t) + \psi^2_\sigma(.,t)\|_\infty^{K\times[0,1]}$$

where K is a compact that contains all the trajectory starting from any unit norm initial condition. Since $\forall t \in \mathbb{R}_+$, $\|\partial_s \psi_\sigma(s,t)\| \leq C$ and $\|\psi_\sigma(s,t)^2\| \leq \frac{1}{2}$, there exists $\tilde{C} > 0$ and independent of t such that

$$\|x(\frac{n}{\sigma(N)}) - y_n\| \leq \frac{\tilde{C}}{\sigma(N)}$$

Now, let $e_n = y_n - x_n$. We have

$$e_{n+1} = e_n(1 + \frac{1}{\sigma(N)}\psi_\sigma(\frac{n}{\sigma(N)}, t)) + \frac{1}{\sigma(N)}(\psi_\sigma(\frac{n}{\sigma(N)}, t) - \psi_{\sigma(N)}(\frac{n}{\sigma(N)}, t))x_n.$$

Since $\|\theta_n^N\| \leq \frac{1}{2}$ and x_0 has unit norm, there exists $M > 0$ independent of x_0 such that, $\forall n$ and N, $\|x_n\| \leq M$. Thus

$$\|e_{n+1}\| \leq \|e_n\|(1 + \frac{1}{2\sigma(N)}) + \frac{1}{\sigma(N)} \sup_{(s,t)\in[0,1]\times\mathbb{R}_+} \|\psi_\sigma(s,t) - \psi_{\sigma(N)}(s,t)\|M.$$

The fact that $\sup_{(s,t)\in[0,1]\times\mathbb{R}_+} \|\psi_\sigma(s,t) - \psi_{\sigma(N)}(s,t)\| \to 0$ (uniform convergence of $\psi_{\sigma(N)}$ to ψ_σ) along with the discrete Gronwall's lemma leads to $\|e_n\| = o(1)$ independent of t and x_0. More precisely,

$$\sup_{t\in\mathbb{R}_+, x_0\in\mathbb{R}^d, \|x_0\|=1} \|\Pi^{\sigma(N)}(t)x_0 - \Pi(t)x_0\| \to 0$$

as $N \to \infty$. We obtain the uniform convergence with t.

□

We can now prove our Th. 2.5.

Proof. Consider $(\psi_{\sigma(N)})_N$ a sub-sequence of $(\psi_N)_N$ as in lemma 2.6 that converges to some ψ_σ.

1) We first prove the uniqueness of the limit.

We want to show that ψ_σ does not depend on σ. This will imply the uniqueness of any accumulation point of the relatively compact sequence $(\psi_N)_N$ and thus its convergence.

We have $\forall s \in [0,1]$,

$$\partial_t \psi_{\sigma(N)}(s,t) = -\Pi^{\sigma(N)\top}_{:\lfloor\sigma(N)s\rfloor}(t)(\Pi^{\sigma(N)}(t) - B)\Pi^{\sigma(N)\top}_{\lfloor\sigma(N)s\rfloor:}(t).$$

As $N \to \infty$, we have thanks to lemma 2.13 that the right hand term converges uniformly to

$$-\Pi^\top_{:s}(t)(\Pi(t) - B)\Pi^\top_{s:}(t)$$

where Π maps x_0 to the solution at time 1 of the Neural ODE $\frac{\mathrm{d}x}{\mathrm{d}s} = \psi_\sigma(s,t)x(s)$ with initial condition x_0, $\Pi_{:s}(t)$ maps x_0 to the solution at time s of the Neural ODE $\frac{\mathrm{d}x}{\mathrm{d}s} = \psi_\sigma(s,t)x(s)$ with initial condition x_0 and $\Pi_{s:}(t)$ maps x_0 to the solution at time $1-s$ of the Neural ODE $\frac{\mathrm{d}x}{\mathrm{d}s} = \psi_\sigma(s,t)x(s)$ with initial condition x_0.

This uniform convergence makes it possible to consider the limit ODE as $N \to \infty$:

$$\partial_t \psi_\sigma(.,t) = F(\psi_\sigma(.,t)), \quad \psi_\sigma(.,0) = 0_{d\times d} \tag{2.11}$$

where $\forall s \in [0,1]$,

$$F(\psi_\sigma(s,t)) = -\Pi^\top_{:s}(t)(\Pi(t) - B)\Pi^\top_{s:}(t).$$

We now show that F is Lipschitz continuous which will guarantee uniqueness through the Picard–Lindelöf theorem. Recall that we have $\forall (s,t) \in [0,1] \times \mathbb{R}_+$:

$$\|\psi_\sigma(s,t)\| \leq \frac{1}{2}.$$

Let ψ_1, ψ_2 with $\|\psi_1(s,t)\| \leq \frac{1}{2}$ and $\|\psi_2(s,t)\| \leq \frac{1}{2}$ and $\Pi_1(t)$, $\Pi_2(t)$ the corresponding flows.

Let x_0 in $\mathbb{R}^d$ with unit norm, x_1 (resp. x_2) be the solutions of $\frac{\mathrm{d}x}{\mathrm{d}s} = \psi_1(s,t)x(s)$ (resp. $\frac{\mathrm{d}x}{\mathrm{d}s} = \psi_2(s,t)x(s)$) with initial condition x_0. Let $y = x_1 - x_2$.

One has $\Pi_1(t)x_0 = x_1(1)$ and $\Pi_2(t)x_0 = x_2(1)$. One has $\dot{y} = \psi_1 x_1 - \psi_2 x_2 = \psi_2 y + (\psi_1 - \psi_2)x_1$. Hence, since $y(0) = 0$, $\|y(s)\| \leq \int_0^s \|\psi_2\|\|y\| + \|\psi_1 - \psi_2\|_\infty \|x_1\|_\infty$, we have

$$\|y(s)\| \leq \frac{1}{2}\|y(s)\| + \|\psi_1 - \psi_2\|_\infty . \|\Pi_1(t)\|$$

and since $\forall t \in \mathbb{R}_+$, $\|\Pi_1(t)\| \leq 2e$ we get

$$\|\Pi_1(t)x_0 - \Pi_2(t)x_0\| = \|y(1)\| \leq \alpha \|\psi_1 - \psi_2\|_\infty$$

for some $\alpha > 0$. The same arguments go for $\Pi_{:s}$ and $\Pi_{s:}$.

Since we only consider maps ψ_σ such that $\|\psi_\sigma(s,t)\| \leq \frac{1}{2}$, this implies that the product is also Lipschitz and thus F is Lipschitz. This guarantees the uniqueness of a solution ψ to the Cauchy problem and we have that $\psi_N \to \psi$ uniformly.

2) We now turn to the convergence speed.

We have $\|\psi_{2N} - \psi_N\| \leq \frac{D}{N}$ for some $D > 0$ thanks to lemma 2.7. For $k \in \mathbb{N}$, we have that

$$\|\psi_{2^k N} - \psi_N\| \leq \sum_{i=0}^{k-1} \|\psi_{2^{i+1}N} - \psi_{2^i N}\| \leq \frac{D}{N} \sum_{i=0}^{k-1} \frac{1}{2^i} \leq \frac{2D}{N}.$$

Letting $k \to \infty$ finally gives $\|\psi - \psi_N\| \leq \frac{2D}{N}$.

□

2.A.6 Proof of Prop. 2.9

Proof. We denote $r_n = \tilde{x}_n - x_n$.

One has $r_N = 0$ and

$$r_n = \tilde{x}_{n+1} - \frac{1}{N} f(\tilde{x}_{n+1}, \theta_n^N) - x_{n+1} + \frac{1}{N} f(x_n, \theta_n^N),$$

that is

$$r_n = r_{n+1} + \frac{1}{N}(f(x_{n+1} - \frac{1}{N} f(x_n, \theta_n^N), \theta_n^N) - f(\tilde{x}_{n+1}, \theta_n^N)).$$

Since

$$f(x_{n+1} - \frac{1}{N} f(x_n, \theta_n^N), \theta_n^N) = f(x_{n+1}, \theta_n^N) - \frac{1}{N} \partial_x f(x_{n+1}, \theta_n^N)[f(x_n, \theta_n^N)] + O(\frac{1}{N^2})$$

this gives

$$r_n = r_{n+1} + \frac{1}{N}(f(x_{n+1}, \theta_n^N) - f(\tilde{x}_{n+1}, \theta_n^N)) - \frac{1}{N^2} \partial_x f(x_{n+1}, \theta_n^N)[f(x_n, \theta_n^N)] + O(\frac{1}{N^3}).$$

Denoting

$$K_N = \sup_{n\in[N-1]} \|\partial_x f(.,\theta_n^N)\|_\infty^K \|f(.,\theta_n^N)\|_\infty^K,$$

we have the following inequality:

$$\|r_n\| \leq (1+\frac{L_f}{N})\|r_{n+1}\| + \frac{1}{N^2}K_N + O(\frac{1}{N^3})$$

and since $r_N = 0$, the discrete Gronwall lemma leads to $\|r_n\| \leq \frac{e^{L_f}-1}{L_f N}K_N + O(\frac{1}{N^2})$. In addition, one has $K_N \leq L_f C_f$ so that

$$\|r_n\| \leq \frac{e^{L_f}-1}{N}C_f + O(\frac{1}{N^2}).$$

□

2.A.7 Proof of Prop. 2.10

Proof. 1) We first control the error made in the gradient with respect to activations.

Denote

$$g_n = \nabla_{\tilde{x}_n} L - \nabla_{x_n} L.$$

One has using formulas (2.3) and (2.7) that

$$g_n = g_{n+1} + \frac{1}{N}(\partial_x f(\tilde{x}_n,\theta_n^N) - \partial_x f(x_n,\theta_n^N))^\top \nabla_{\tilde{x}_{n+1}} L + \frac{1}{N}\partial_x f(x_n,\theta_n^N)^\top g_{n+1}.$$

Since

$$\|\partial_x f(x_n,\theta_n^N)^\top g_{n+1}\| \leq L_f \|g_{n+1}\|$$

and because

$$\|(\partial_x f(\tilde{x}_n,\theta_n^N) - \partial_x f(x_n,\theta_n^N))^\top \nabla_{\tilde{x}_{n+1}} L\| \leq L_{df}\|\tilde{x}_n - x_n\| g,$$

where g is a bound on $\nabla_{\tilde{x}_{n+1}} L$, we conclude by using Prop. 2.9 and the discrete Gronwall's lemma.

2) We can now control the gradients with respect to the parameters θ_n^N's.

Denote

$$t_n = \tilde{\nabla}_{\theta_n^N} L - \nabla_{\theta_n^N} L.$$

We have

$$N t_n = -[\partial_\theta f(x_n,\theta_n^N) - \partial_\theta f(\tilde{x}_n,\theta_n^N)]^\top \nabla_{x_n} L - [\partial_\theta f(\tilde{x}_n,\theta_n^N)]^\top g_n.$$

Hence $N\|t_n\| \leq L_\theta \|x_n - \tilde{x}_n\| g + C_\theta \|g_n\|$ where g is a bound on $\nabla_{x_n} L$.

Using our bound on $\|g_n\|$ and Prop. 2.9 we get

$$N\|t_n\| \leq \frac{L_\theta(e^{L_f}-1)gC_f}{N} + \frac{(e^{L_f}-1)(e^{L_f}-1)C_f L_{df} g C_\theta}{L_f N} + O(\frac{1}{N^2})$$

and thus

$$t_n = O(\frac{1}{N^2}).$$

□

2.A.8 Proof of Prop. 2.11

In the following, we let for short $f_n(x) = f(x, \theta_n^N)$, and we define

$$\varphi_n(x) = \frac{1}{2}\left(f_n(x) + f_{n+1}(x + \frac{1}{N}f_n(x))\right) \text{ and } \psi_n(x) = \frac{1}{2}\left(f_{n+1}(x) + f_n(x - \frac{1}{N}f_{n+1}(x))\right) \tag{2.12}$$

so that Heun's forward and backward equations are

$$x_{n+1} = x_n + \frac{1}{N}\varphi_n(x_n) \text{ and } \tilde{x}_n = \tilde{x}_{n+1} - \frac{1}{N}\psi_n(\tilde{x}_{n+1}).$$

We have the following lemma that quantifies the reconstruction error over one iteration:

Lemma 2.14. *For $x \in \mathbb{R}$, we have as N goes to infinity*

$$\psi_n(x + \frac{1}{N}\varphi_n(x)) - \varphi_n(x) = \frac{1}{4N}\left(J_{n+1}(x) - J_n(x)\right)\left[f_{n+1}(x) - f_n(x)\right] + O(\frac{1}{N^2}),$$

where $J_n = \partial_x f_n(x)$ is the Jacobian of f_n.

Proof. As N goes to infinity, we have the following expansions of (2.12):

$$\varphi_n(x) = \frac{1}{2}(f_n(x) + f_{n+1}(x)) + \frac{1}{2N}J_{n+1}(x)[f_n(x)] + O(\frac{1}{N^2}),$$

$$\psi_n(x) = \frac{1}{2}(f_n(x) + f_{n+1}(x)) - \frac{1}{2N}J_n(x)[f_{n+1}(x)] + O(\frac{1}{N^2}).$$

As a consequence, we have

$$\begin{aligned}\psi_n(x + \frac{1}{N}\varphi_n(x)) =& \frac{1}{2}(f_n(x) + f_{n+1}(x)) - \frac{1}{2N}J_n(x)[f_{n+1}(x)] \\ &+ \frac{1}{4N}(J_n(x)[f_n(x) + f_{n+1}(x)] + J_{n+1}(x)[f_n(x) + f_{n+1}(x)]) + O(\frac{1}{N^2}).\end{aligned}$$

Putting everything together, we find that the zero-th order in $\psi_n(x + \frac{1}{N}\varphi_n(x)) - \varphi_n(x)$ cancels, and that the first order simplifies to $\frac{1}{4N}\left(J_{n+1}(x) - J_n(x)\right)\left[f_{n+1}(x) - f_n(x)\right]$. □

We now turn the the proof of the main proposition:

Proof. We let $r_n = \tilde{x}_n - x_n$ the reconstruction error. We have $r_N = 0$, and we find

$$\begin{aligned} r_n &= \tilde{x}_n - x_n && (2.13)\\ &= \tilde{x}_{n+1} - \frac{1}{N}\psi_n(\tilde{x}_{n+1}) - x_{n+1} + \frac{1}{N}\varphi_n(x_n) && (2.14)\\ &= r_{n+1} - \frac{1}{N}\left(\psi_n(\tilde{x}_{n+1}) - \psi_n(x_{n+1})\right) - \frac{1}{N}\left(\psi_n(x_{n+1}) - \varphi_n(x_n)\right). && (2.15)\end{aligned}$$

Using the triangle inequality, and the L_f'−Lispchitz continuity of ψ_n, we get

$$\|r_n\| \leq (1 + \frac{L_f'}{N})\|r_{n+1}\| + \frac{1}{N}\|\psi_n(x_{n+1}) - \varphi_n(x_n)\|.$$

The last term is controlled with the previous Lemma 2.14:

$$\|\psi_n(x_{n+1}) - \varphi_n(x_n)\| \leq \frac{1}{4N}\| \left(J_{n+1}(x_n) - J_n(x_n)\right) \left[f_{n+1}(x_n) - f_n(x_n)\right]\| + O(\frac{1}{N^2}) \tag{2.16}$$

$$\leq \frac{C'_f \Delta_\theta^N}{N} + O(\frac{1}{N^2}). \tag{2.17}$$

We therefore get the recursion

$$\|r_n\| \leq (1 + \frac{L'_f}{N})\|r_{n+1}\| + \frac{C'_f \Delta_\theta^N}{N^2} + O(\frac{1}{N^3}).$$

Unrolling the recursion gives,

$$\|r_n\| \leq \frac{(e^{L'_f} - 1)C'_f}{L'_f N}\Delta_\theta^N + O(\frac{1}{N^2}).$$

□

2.A.9 Proof of Prop. 2.12

Proof. 1) We first control the error made in the gradient with respect to activations. We have the following recursions:

$$\nabla_{x_n} L = (I + \frac{1}{N}\partial_x \varphi_n(x_{n+1}))^\top \nabla_{x_{n+1}} L \text{ and } \nabla_{\tilde{x}_n} L = (I + \frac{1}{N}\partial_x \varphi_n(\tilde{x}_{n+1}))^\top \nabla_{\tilde{x}_{n+1}} L$$

Letting $r'_n = \nabla_{x_n} L - \nabla_{\tilde{x}_n} L$, we have

$$r'_n = r'_{n+1} + \frac{1}{N}\partial_x \varphi_n(x_{n+1})^\top r'_{n+1} + \frac{1}{N}\left(\partial_x \varphi_n(x_{n+1}) - \partial_x \varphi_n(\tilde{x}_{n+1})\right)^\top \nabla_{\tilde{x}_{n+1}} L$$

Therefore, using the triangle inequality, and letting g a bound on the norm of the gradients $\nabla_{\tilde{x}_{n+1}} L$ and Δ a Lipschitz constant of $\partial_x \varphi_n$, we find

$$\|r'_n\| \leq (1 + \frac{L'_f}{N})\|r'_{n+1}\| + \frac{1}{N} g\Delta \|x_{n+1} - \tilde{x}_{n+1}\|$$

The last term is controled with the previous proposition, and we find

$$\|r'_n\| \leq (1 + \frac{L'_f}{N})\|r'_{n+1}\| + \frac{(e^{L'_f} - 1)C'_f g\Delta}{L'_f N^2}\Delta_\theta^N + O(\frac{1}{N^3}),$$

which gives by unrolling:

$$\|r'_n\| \leq \frac{(e^{L'_f} - 1)^2 C'_f g\Delta}{L'^2_f N}\Delta_\theta^N + O(\frac{1}{N^2}).$$

2) We can now control the gradients with respect to parameters. Since Heun's method involves parameters θ_n^N both for the computation of x_n and x_{n+1}, the gradient formula is slightly more complicated than for the classical ResNet. It is the sum of two terms, the first one $\nabla^1_{\theta_n^N} L$ corresponding to iteration n and the second one $\nabla^2_{\theta_n^N} L$ corresponding to iteration $n-1$.

We have

$$\nabla^1_{\theta^N_n} L = \frac{1}{2N}\left(\partial_\theta f(x_n, \theta^N_n) + \frac{1}{N}\partial_x f(y_n, \theta^N_{n+1})\partial_\theta f(x_n, \theta^N_n)\right)^\top \nabla_{x_n} L$$

and

$$\nabla^2_{\theta^N_n} L = \frac{1}{2N}\left(\partial_\theta f(y_{n-1}, \theta^N_{n-1})\right)^\top (I + \frac{1}{N}\partial_x f(x_{n-1}, \theta^N_{n-1}))^\top \nabla_{x_{n-1}} L.$$

The gradient $\nabla_{\theta^N_n} L$ is finally

$$\nabla_{\theta^N_n} L = \nabla^1_{\theta^N_n} L + \nabla^2_{\theta^N_n} L.$$

Overall, these equations map the activations x_n and x_{n-1}, and the gradients $\nabla_{x_{n-1}} L$ and $\nabla_{x_n} L$ to the gradient $\nabla_{\theta^N_n}$, which we rewrite as

$$\nabla_{\theta^N_n} L = \Psi(x_n, x_{n-1}, \nabla_{x_n} L, \nabla_{x_{n-1}} L),$$

where the function Ψ is explicitly defined by the above equations. With the memory-free backward pass, the gradient is rather estimated as

$$\tilde{\nabla}_{\theta^N_n} L = \Psi(\tilde{x}_n, \tilde{x}_{n-1}, \nabla_{\tilde{x}_n} L, \nabla_{\tilde{x}_{n-1}} L).$$

The function Ψ is Lispchitz-continuous since all functions involved in its composition are Lipschitz-continuous and the activations belong to a compact set, and its Lipschitz constant scales as $\frac{1}{N}$. We write its Lipschitz constant as $\frac{L_\Psi}{N}$, and we get:

$$\|\nabla_{\theta^N_n} L - \tilde{\nabla}_{\theta^N_n} L\| = \|\Psi(x_n, x_{n-1}, \nabla_{x_n} L, \nabla_{x_{n-1}} L) - \Psi(\tilde{x}_n, \tilde{x}_{n-1}, \nabla_{\tilde{x}_n} L, \nabla_{\tilde{x}_{n-1}} L)\| \tag{2.18}$$

$$\leq \frac{L_\Psi}{N}(\|x_n - \tilde{x}_n\| + \|x_{n-1} - \tilde{x}_{n-1}\| + \|\nabla_{x_n} L - \nabla_{\tilde{x}_n} L\| + \|\nabla_{x_{n-1}} L - \nabla_{\tilde{x}_{n-1}} L\|). \tag{2.19}$$

Using the previous propositions, we get:

$$\|\nabla_{\theta^N_n} L - \tilde{\nabla}_{\theta^N_n} L\| = O(\frac{\Delta^N_\theta}{N^2} + \frac{1}{N^3}).$$

□

2.B Experimental details

In all our experiments, we use Nvidia Tesla V100 GPUs.

2.B.1 CIFAR

For our experiments on CIFAR-10 (training from scratch), we used a batch-size of 128 and we employed SGD with a momentum of 0.9. The training was done over 200 epochs. The initial learning rate was 0.1 and we used a cosine learning rate scheduler. A constant weight decay was set to 5×10^{-4}. Standard inputs preprocessing as proposed in Pytorch (Paszke et al., 2017) was performed.

For our finetuning experiment on CIFAR-10, we used a batch-size of 128 and we employed SGD with a momentum of 0.9. The training was done over 5 epochs. The learning rate was kept constant to 10^{-3}. A constant weight decay was set to 5×10^{-4}. Standard inputs preprocessing as proposed in Pytorch was also performed.

For our experiment with our simple ResNet model that processes the input by a 5×5 convolution with 16 out channels, we used a batch-size of 256 and we employed SGD with a momentum of 0.9. The training was done over 90 epochs. The learning rate was set to 10^{-1} and was decayed by a factor 10 every 30 epochs. A constant weight decay was set to 5×10^{-4}. Standard inputs preprocessing as proposed in Pytorch was also performed.

2.B.2 ImageNet

For our experiments on ImageNet (training from scratch), we used a batch-size of 256 and we employed SGD with a momentum of 0.9. The training was done over 100 epochs. The initial learning rate was 0.1 and was decayed by a factor 10 every 30 epochs. A constant weight decay was set to 10^{-4}. Standard inputs preprocessing as proposed in Pytorch was performed: normalization, random croping of size 224×224 pixels, random horizontal flip.

For our finetuning experiment on ImageNet, we used a batch-size of 256 and we employed SGD with a momentum of 0.9. The training was done over 3 epochs. The learning rate was kept constant to 5×10^{-4}. A constant weight decay was set to 10^{-4}. Standard inputs preprocessing as proposed in Pytorch was performed: normalization, random croping of size 224×224 pixels, random horizontal flip.

2.C Architecture details

In computer vision, the ResNet as presented in (He et al., 2016a) first applies non residual transformations to the input image: a feature extension convolution that goes to 3 channels to 64, a batch norm, a non-linearity (ReLU) and optionally a maxpooling.

It is then made of 4 layers (each layer is a series of residual blocks) of various depth, all of which perform residual connections. Each of the 4 layers works at different scales (with an input with a different number of channels): typically 64, 128, 256 and 512 respectively. There are two types of residual blocks: Basic Blocks and Bottlenecks. Both are made of a successions of convolutions conv, batch normalizations bn (Ioffe and Szegedy, 2015) and ReLU non-linearity σ. For example, a Basic Block iterates (in a pre-activation (He et al., 2016b) fashion):

$$x \to x + \mathrm{bn}(\mathrm{conv}(\sigma(\mathrm{bn}(\mathrm{conv}(\sigma(x)))))).$$

Finally, there is a classification module: average pooling followed by a fully connected layer.

3 Implicit regularization of deep residual networks towards neural ODEs

In this chapter, we take a further step towards establishing a solid mathematical link between residual neural networks and neural ordinary differential equations (ODEs), by proving an implicit regularization of deep residual networks towards neural ODEs. Our result holds for nonlinear networks trained with gradient flow. We prove that if the network is initialized as a discretization of a neural ODE, then such a discretization holds throughout training. Our results are valid for a finite training time, and also as the training time tends to infinity provided that the network satisfies a Polyak-Łojasiewicz condition. Importantly, this condition holds for a family of residual networks where the residuals are two-layer perceptrons with an overparameterization in width that is only linear, and implies the convergence of the gradient flow to a global minimum of the loss. Our results are illustrated by numerical experiments.

Contents

Specific notations. In this chapter, we use the notation h_k (resp. $H(s)$) instead of x_n (resp. $X(s)$) to denote the residual network's activations (resp. neural ODE trajectories). Additionally, L represents the network's depth, in contrast to N as used in Chapter 2 and the Introduction. The parameters are denoted by $\mathcal{Z}$ instead of Θ.

3.1 Introduction

Residual networks are a successful family of deep learning models popularized by breakthrough results in computer vision (He et al., 2016a). The key idea of residual networks, namely the presence of skip connections, is now ubiquitous in deep learning, and can be found, for example, in Transformer models (Vaswani et al., 2017). The main advantage of skip connections is to allow successful training with depth of the order of a thousand layers, in contrast to vanilla neural networks, leading to significant performance improvements (e.g., Wang et al., 2022). This has motivated research on the properties of residual networks in the limit where the depth tends to infinity. One of the main explored directions is the neural ordinary differential equation (ODE) limit (Chen et al., 2018).

To present the neural ODE principle, we first introduce the mathematical formalism of deep residual networks. We consider a single model throughout the chapter to simplify the exposition, but most of our results apply to more general models, as will be discussed later. We consider the formulation

$$h_{k+1} = h_k + \frac{1}{L\sqrt{m}} V_{k+1} \sigma\Big(\frac{1}{\sqrt{q}} W_{k+1} h_k\Big), \quad k \in \{0, \dots, L-1\}, \tag{3.1}$$

where L is the depth of the network, $h_k \in \mathbb{R}^q$ is the output of the k-th hidden layer, $V_k \in \mathbb{R}^{q \times m}$, $W_k \in \mathbb{R}^{m \times q}$ are the weights of the k-th layer, and σ is an activation function applied element-wise. Scaling with the square root of the width is classical, although it often appears as an equivalent condition on the variance at initialization (Glorot and Bengio, 2010; LeCun et al., 2012; He et al., 2015b). We make the scaling factors explicit to have weights of magnitude $\mathcal{O}(1)$ independently of the width and the depth. The $1/L$ scaling factor is less common, but it is necessary for the correspondence with neural ODEs to hold. More precisely, if there exist Lipschitz continuous functions $\mathcal{V}$ and $\mathcal{W}$ such that $V_k = \mathcal{V}(k/L)$ and $W_k = \mathcal{W}(k/L)$, then the residual network (3.1) converges, as $L \to \infty$, to the ODE

$$\frac{dH}{ds}(s) = \frac{1}{\sqrt{m}} \mathcal{V}(s) \sigma\Big(\frac{1}{\sqrt{q}} \mathcal{W}(s) H(s)\Big), \quad s \in [0, 1], \tag{3.2}$$

where s is the continuous-depth version of the layer index. It is important to note that this correspondence holds for *fixed* limiting functions $\mathcal{V}$ and $\mathcal{W}$. This is especially true at initialization, for example by setting the V_k to zero and the W_k to weight-tied Gaussian matrices. In this case, the initial residual network is trivially equal to the neural ODE $\frac{dH}{ds}(s) = 0$. Of course, more sophisticated initialization choices are possible, as shown, e.g., in Marion et al. (2022) and Sander et al. (2022b). However, regardless of an ODE structure at initialization, a more challenging question is that of the structure of the network *after* training. Since the weights are updated during training, there is no a priori guarantee that an ODE limit still holds after training, even if it does at initialization.

The question of a possible ODE structure for the trained network is not a mere technical one. In fact, it is important for at least three reasons. First, it gives a precise answer to the question of the connection between (trained) residual networks and neural ODEs, providing more solid ground to a common statement in the community that both can coincide in the large-depth limit (see, e.g., Haber and Ruthotto, 2017a; E et al., 2019; Dong et al., 2020; Massaroli et al.,

2020; Kidger, 2022). Second, it opens exciting perspectives for understanding residual networks. Indeed, if trained residual networks are discretizations of neural ODEs, then it is possible to apply results from neural ODEs to the large family of residual networks. In particular, from a theoretical point of view, the approximation capabilities of neural ODEs are well understood (Teshima et al., 2020; Zhang et al., 2020a) and it is relatively easy to obtain generalization bounds for these models (see Hanson and Raginsky, 2022 and Chapter Marion, 2023). From a practical standpoint, advantages of neural ODEs include memory-efficient training (Chen et al., 2018; Sander et al., 2022b) and weight compression (Queiruga et al., 2021). This is important because in practice memory is a bottleneck for training residual networks (Gomez et al., 2017). Finally, our analysis is a first step towards understanding the implicit regularization (Neyshabur et al., 2014; Vardi, 2023) of gradient descent for deep residual networks, that is, characterizing the properties of the trained network among all minimizers of the empirical risk.

Throughout the document, it is assumed that the network is trained with gradient flow, which is a continuous analog of gradient descent. The parameters V_k are updated according to an ODE of the form $\frac{dV_k}{dt}(t) = -L\frac{\partial \ell}{\partial V_k}(t)$ for $t \geq 0$, where ℓ is an empirical risk (the exact mathematical context and assumptions are detailed in Section 3.3), and similarly for W_k. The scaling factor L is the counterpart of the factor $1/L$ in (3.1), and prevents exploding or vanishing gradients as L tends to infinity. Note that the gradient flow is defined with respect to a time index t different from the layer index s.

Contributions. Our first main contribution (Section 3.4.1) is to show that a neural ODE limit holds after training up to time t, i.e., there exists a function $\mathcal{V}(s,t)$ such that the residual network converges, as L tends to infinity, to the ODE

$$\frac{dH}{ds}(s) = \frac{1}{\sqrt{m}}\mathcal{V}(s,t)\sigma\Big(\frac{1}{\sqrt{q}}\mathcal{W}(s,t)H(s)\Big), \quad s \in [0,1].$$

This large-depth limit holds for any finite training time $t \geqslant 0$. However, the convergence of the optimization algorithm as t tends to infinity, which we refer to as the *long-time limit* to distinguish it from the large-depth limit $L \to \infty$, is not guaranteed without further assumptions, due to the non-convexity of the optimization problem. We attack the question (Section 3.4.2) when the width is large enough by proving a Polyak-Łojasiewicz (PL) condition, which is now state of the art in analyzing the properties of optimization algorithms for deep neural networks (Liu et al., 2022a). The main assumption for our PL condition to hold is that the width m of the hidden layers should be greater than some constant times the number of data n. As a second main contribution, we show that the PL condition yields the long-time convergence of the gradient flow for residual networks with linear overparameterization. Finally, we prove the convergence with high probability in the long-time limit, namely the existence of functions $\mathcal{V}_\infty$ and $\mathcal{W}_\infty$ such that the discrete trajectory defined by the trained residual network (3.1) converges as *both* L and t tend to infinity to the solution of the neural ODE (3.2) with $\mathcal{V} = \mathcal{V}_\infty$ and $\mathcal{W} = \mathcal{W}_\infty$. In addition, our approach points out that this limiting ODE interpolates the training data. Finally, our results are illustrated by numerical experiments (Section 3.5).

Organization of the chapter. Section 3.2 presents some related work. We then move on to detail the mathematical context and notation in Section 3.3 before giving our main results in Section 3.4. Section 3.5 is devoted to numerical experiments. We conclude the main part of the chapter in Section 3.6. Then, in Section 3.A, we prove results on a more general residual network model that encompasses the one presented so far. These results are then instantiated in the specific case of the residual network (3.3) in Section 3.B, thus proving the results of the main

part of the chapter. Section 3.C contains some lemmas that are useful for the proofs. We present in Section 3.D a counter-example showing that a residual network with the ReLU activation can move away from the neural ODE structure during training. Finally, Section 3.E presents some experimental details.

3.2 Related work

Deep residual networks and neural ODEs. Several works study the large-depth convergence of residual networks to differential equations, but without considering the training dynamics, such as Cohen et al. (2021); Thorpe and van Gennip (2022); Hayou (2023); Marion et al. (2022). Closer to our setting, Cont et al. (2022) and Sander et al. (2022b) analyze the dynamics of gradient descent for deep residual networks, as we do, but with significant differences. Cont et al. (2022) consider a $1/\sqrt{L}$ scaling factor in front of the residual branch, resulting in a limit that is not a neural ODE. In addition, only W is trained. Furthermore, to obtain convergence in the long-time limit, it is assumed that the data points are nearly orthogonal. Sander et al. (2022b) prove the existence of an ODE limit for trained residual networks, but in the simplified case of a linear activation and under a more restricted setting.

Long-time convergence of wide residual networks. Polyak-Łojasiewicz conditions are a modern tool to prove long-time convergence of overparameterized neural networks (Liu et al., 2022a). These conditions are a relaxation of convexity, and mean that the gradients of the loss with respect to the parameters cannot be small when the loss is large. They have been applied to residual networks with both linear (Bartlett et al., 2018; Wu et al., 2019; Zou et al., 2020b) and nonlinear activations (Allen-Zhu et al., 2019; Frei et al., 2019; Barboni et al., 2022; Cont et al., 2022; MacDonald et al., 2022). Building on the proof technique of Nguyen and Mondelli (2020) for non-residual networks, we need only a linear overparameterization to prove our PL condition, i.e., we require $m = \Omega(n)$. This compares favorably with results requiring polynomial overparameterization (Allen-Zhu et al., 2019; Barboni et al., 2022) or assumptions on the data, either a margin condition (Frei et al., 2019) or a sample size smaller than the dimension of the data space (Cont et al., 2022; MacDonald et al., 2022).

Implicit regularization. This chapter can be related to a line of work on the implicit regularization of gradient-based algorithms for residual networks (Neyshabur et al., 2014). We show that the optimization algorithm does not just converge to any residual network that minimizes the empirical risk, but rather to the discretization of a neural ODE. Note that most implicit regularization results state that the optimization algorithm converges to an interpolator that minimizes some complexity measure, which can be a margin (Lyu and Li, 2020), a norm (Boursier et al., 2022), or a matrix rank (Li et al., 2021). Thus, an interesting next step is to understand if the neural ODE found by gradient flow actually minimizes some complexity measure, and to characterize its generalization properties.

3.3 Definitions and notation

This section is devoted to specifying the setup outlined in Section 3.1.

Residual network. A (scaled) residual network of depth $L \in \mathbb{N}^*$ is defined by

$$\begin{aligned} h_0^L &= A^L x \\ h_{k+1}^L &= h_k^L + \frac{1}{L\sqrt{m}} V_{k+1}^L \sigma\Big(\frac{1}{\sqrt{q}} W_{k+1}^L h_k^L\Big), \quad k \in \{0, \dots, L-1\}, \\ F^L(x) &= B^L h_L^L. \end{aligned} \tag{3.3}$$

To allow the hidden layers $h_k^L \in \mathbb{R}^q$ to have a different dimension than the input $x \in \mathbb{R}^d$, we first map x to h_0^L with a weight matrix $A^L \in \mathbb{R}^{q \times d}$. We assume that the hidden layers belong to a higher dimensional space than the input and output, i.e., $q \geqslant \max(d, d')$. The residual transformations are two-layer perceptrons parameterized by the weight matrices $V_k^L \in \mathbb{R}^{q \times m}$ and $W_k^L \in \mathbb{R}^{m \times q}$. This is standard in the literature (e.g., He et al., 2016b; Chen et al., 2018; Teh et al., 2019; Barboni et al., 2022). The last weight matrix $B^L \in \mathbb{R}^{d' \times q}$ maps the last hidden layer to the output $F^L(x)$ in $\mathbb{R}^{d'}$. Also, $\sigma : \mathbb{R} \to \mathbb{R}$ is an element-wise activation function assumed to be $\mathcal{C}^2$, non-constant, Lipschitz continuous, bounded, and such that $\sigma(0) = 0$. The convenient shorthand $Z_k^L = (V_k^L, W_k^L)$ is occasionally used, and we denote $\|Z_k^L\|_F$ the sum of the Frobenius norms $\|V_k^L\|_F + \|W_k^L\|_F$.

Data and loss. The data is a sample of n pairs $(x_i, y_i)_{1 \leq i \leq n} \in (\mathcal{X} \times \mathcal{Y})^n$ where $\mathcal{X} \times \mathcal{Y}$ is a compact set of $\mathbb{R}^d \times \mathbb{R}^{d'}$. The empirical risk is the mean squared error $\ell^L = \frac{1}{n} \sum_{i=1}^n \|F^L(x_i) - y_i\|^2$.

Initialization. We initialize $A^L = (I_{\mathbb{R}^{d \times d}}, 0_{\mathbb{R}^{(q-d) \times d}})$ as the identity matrix in $\mathbb{R}^{d \times d}$ concatenated row-wise with the zero matrix in $\mathbb{R}^{(q-d) \times d}$, to act as a simple projection of the input onto the higher dimensional space $\mathbb{R}^q$, and similarly $B^L = (0_{\mathbb{R}^{d' \times (q-d')}}, I_{\mathbb{R}^{d' \times d'}})$. The weights V_k^L are initialized to zero and the W_k^L as weight-tied standard Gaussian matrices, i.e., for all $k \in \{1, \dots, L\}$, $W_k^L = W \sim \mathcal{N}(0,1)^{\otimes (m \times q)}$. Initializing outer matrices to zero is standard practice (Zhang et al., 2019a), while taking weight-tied matrices instead of i.i.d. ones is less common. We show in Section 3.5 that it is still possible to learn with this initialization scheme on real world data. As explained in Section 3.4.3, other initialization choices are possible, provided they correspond to the discretization of a Lipschitz continuous function, but we focus on this one in the main text for simplicity.

Training algorithm. Gradient flow is the limit of gradient descent as the learning rate tends to zero. The parameters are set at time $t = 0$ by the initialization, and then evolve according to the ODE

$$\frac{dA^L}{dt}(t) = -\frac{\partial \ell^L}{\partial A^L}(t), \quad \frac{dZ_k^L}{dt}(t) = -L\frac{\partial \ell^L}{\partial Z_k^L}(t), \quad \frac{dB^L}{dt}(t) = -\frac{\partial \ell^L}{\partial B^L}(t), \quad t \geqslant 0, \tag{3.4}$$

for $k \in \{1, \dots, L\}$. In the following, the dependence in t is made explicit when necessary, e.g., we write $h_k^L(t)$ instead of h_k^L, and $F^L(x; t)$ instead of $F^L(x)$.

It turns out that, without further assumptions, the gradient flow can diverge in finite time. This is because the dynamics are not (globally) Lipschitz continuous, breaking the conditions of the Picard-Lindelöf theorem (see Lemma 3.19) for existence and uniqueness of ODE solutions. A common practice (Goodfellow et al., 2016a, Section 10.11.1) is to consider instead a clipped gradient flow

$$\frac{dA^L}{dt}(t) = \pi\Big(-\frac{\partial \ell^L}{\partial A^L}(t)\Big), \quad \frac{dZ_k^L}{dt}(t) = \pi\Big(-L\frac{\partial \ell^L}{\partial Z_k^L}(t)\Big), \quad \frac{dB^L}{dt}(t) = \pi\Big(-\frac{\partial \ell^L}{\partial B^L}(t)\Big), \tag{3.5}$$

where π is a generic notation for a bounded Lipschitz continuous operator. For example, clipping each coordinate of the gradient at some $C > 0$ amounts to taking π as the projection on the ball centered at 0 of radius C for the ℓ_∞ norm. Clipping ensures that the dynamics are well defined, as shown in the next proposition that is a consequence of the Picard-Lindelöf theorem.

Proposition 3.1. *The (clipped) gradient flow (3.5) has a unique solution for all $t \geqslant 0$.*

Proof. See Section 3.B.1. □

In Section 3.4.2, we make additional assumptions to prove the long-time convergence of the gradient flow. We then prove that these assumptions ensure that the dynamics of the gradient flow (3.4) are well defined, eliminating the need for clipping.

Neural ODE. The neural ODE corresponding to the residual network (3.3) is defined by

$$\begin{aligned} H(0) &= Ax \\ \frac{dH}{ds}(s) &= \frac{1}{\sqrt{m}}\mathcal{V}(s)\sigma\Big(\frac{1}{\sqrt{q}}\mathcal{W}(s)H(s)\Big), \quad s \in [0,1], \\ F(x) &= BH(1), \end{aligned} \tag{3.6}$$

where $x \in \mathbb{R}^d$ is the input, $H \in \mathbb{R}^q$ is the variable of the ODE, $\mathcal{V} : [0,1] \to \mathbb{R}^{q\times m}$ and $\mathcal{W} : [0,1] \to \mathbb{R}^{m\times q}$ are Lipschitz continuous functions, $A \in \mathbb{R}^{q\times d}$ and $B \in \mathbb{R}^{d'\times q}$ are matrices, and the output is $F(x) \in \mathbb{R}^{d'}$. The following proposition shows that the neural ODE is well defined. In addition, its output is close to the residual network (3.3) provided the weights are discretizations of $\mathcal{V}$ and $\mathcal{W}$.

Proposition 3.2. *The neural ODE (3.6) has a unique solution $H : [0,1] \to \mathbb{R}^q$. Consider, moreover, the residual network (3.3) with $A^L = A$, $V_k^L = \mathcal{V}(k/L)$ and $W_k^L = \mathcal{W}(k/L)$ for $k \in \{1,\dots,L\}$, and $B^L = B$. Then there exists $C > 0$ such that, for all $L \in \mathbb{N}^*$, $\sup_{x\in\mathcal{X}} \|F(x) - F^L(x)\| \leq \frac{C}{L}$.*

Proof. See Section 3.B.2. □

Clearly, our choices of V_k^L and W_k^L at initialization are discretizations of the Lipschitz continuous (in fact, constant) functions $\mathcal{V}(s) \equiv 0$ and $\mathcal{W}(s) \equiv W \sim \mathcal{N}(0,1)^{\otimes(m\times q)}$. Thus, Proposition 3.2 holds at initialization, and the residual network is equivalent to the trivial ODE $\frac{dH}{ds}(s) = 0$. The next section shows that after training we obtain non-trivial dynamics, which still discretize neural ODEs.

3.4 Large-depth limit of residual networks

We study the large-depth limit of trained residual networks in two settings. In Section 3.4.1, we consider the case of a finite training time. We move in Section 3.4.2 to the case where the training time tends to infinity, which is tractable under a Polyak-Łojasiewicz condition.

3.4.1 Clipped gradient flow and finite training time

We first consider the case where the neural network is trained with clipped gradient flow (3.5) on some training time interval $[0,T]$, $T > 0$. This allows us to prove large-depth convergence to a neural ODE without further assumptions. We emphasize that stopping training after a finite

training time is a common technique in practice, referred to as early stopping (Goodfellow et al., 2016a, Section 7.8). It is considered as a form of implicit regularization, and our result sheds light on this intuition by showing that the complexity of the trained networks increases with T.

The following proposition is a key step in proving the main theorem of this section.

Proposition 3.3. *There exist $M, K > 0$ such that, for any $t \in [0,T]$, $L \in \mathbb{N}^*$, and $k \in \{1,\dots,L\}$,*

$$\max\left(\left\|A^L(t)\right\|_F, \left\|V_k^L(t)\right\|_F, \left\|W_k^L(t)\right\|_F, \left\|B^L(t)\right\|_F\right) \leq M,$$

and, for $k \in \{1,\dots,L-1\}$,

$$\max\left(\left\|V_{k+1}^L(t) - V_k^L(t)\right\|_F, \left\|W_{k+1}^L(t) - W_k^L(t)\right\|_F\right) \leq \frac{K}{L}.$$

Moreover, with probability at least $1 - \exp\left(-\frac{3qm}{16}\right)$, the following expressions hold for M and K:

$$M = TM_\pi + 2\sqrt{qm}, \quad K = \beta T e^{\alpha T}, \tag{3.7}$$

where M_π is the supremum of π in Frobenius norm, and α and β depend on $\mathcal{X}$, $\mathcal{Y}$, M, and σ.

Proof. See Section 3.B.3. □

This proposition ensures that the size of the weights and the difference between successive weights remain bounded throughout training. We can now state the main result, which states the convergence, for any training time in $[0,T]$, of the neural network to a neural ODE as $L \to \infty$. Recall that a sequence of functions f^L of some variable u is said to converge uniformly over $u \in U$ to f if $\sup_{u\in U} \|f^L(u) - f(u)\| \to 0$.

Theorem 3.4. *Consider the residual network (3.3) with the training dynamics (3.5). Then the following statements hold **as L tends to infinity**:*

(i) There exist functions $A : [0,T] \to \mathbb{R}^{q\times d}$ and $B : [0,T] \to \mathbb{R}^{d'\times q}$ such that $A^L(t)$ and $B^L(t)$ converge uniformly over $t \in [0,T]$ to $A(t)$ and $B(t)$.

(ii) There exists a Lipschitz continuous function $\mathcal{Z} : [0,1] \times [0,T] \to \mathbb{R}^{q\times m} \times \mathbb{R}^{m\times q}$ such that

$$\mathcal{Z}^L : [0,1] \times [0,T] \to \mathbb{R}^{q\times m} \times \mathbb{R}^{m\times q}, \ (s,t) \mapsto \mathcal{Z}^L(s,t) = Z^L_{\lfloor (L-1)s\rfloor+1}(t) \tag{3.8}$$

converges uniformly over $s \in [0,1]$ and $t \in [0,T]$ to $\mathcal{Z} = (\mathcal{V}, \mathcal{W})$.

(iii) Uniformly over $s \in [0,1]$, $t \in [0,T]$, and $x \in \mathcal{X}$, the hidden layer $h^L_{\lfloor Ls\rfloor}(t)$ converges to the solution at time s of the neural ODE

$$\begin{aligned} H(0,t) &= A(t)x \\ \frac{\partial H}{\partial s}(s,t) &= \frac{1}{\sqrt{m}}\mathcal{V}(s,t)\sigma\Big(\frac{1}{\sqrt{q}}\mathcal{W}(s,t)H(s,t)\Big), \quad s \in [0,1]. \end{aligned} \tag{3.9}$$

(iv) Uniformly over $t \in [0,T]$ and $x \in \mathcal{X}$, the output $F^L(x;t)$ converges to $B(t)H(1,t)$.

Proof. See Section 3.B.4. □

Let us sketch the proof of statement (ii), which is the cornerstone of the theorem. A first key idea is to introduce in (3.8) the piecewise-constant continuous-depth interpolation $\mathcal{Z}^L$ of the weights, whose ambient space does not depend on L, in contrast to the discrete weight sequence Z_k^L. Since the weights remain bounded during training by Proposition 3.3, the Arzelà-Ascoli theorem guarantees the existence of an accumulation point for $\mathcal{Z}^L$. We show that the accumulation point is unique because it is the solution of an ODE satisfying the conditions of the Picard-Lindelöf theorem. The uniqueness of the accumulation point then implies the existence of a limit for the weights.

There are two notable byproducts of our proof. The first one is an explicit description of the training dynamics of the limiting weights A, B, and $\mathcal{Z}$, as the solution of an ODE system, as presented in Appendix 3.A.5. The second one, which we now describe, consists of norm bounds on the weights. Proposition 3.3 bounds the discrete weights and the difference between two consecutive weights respectively by some $M, K > 0$. The proof of Theorem 3.4 shows that this bound carries over to the continuous weights, in the sense that $A(t)$, $\mathcal{V}(s,t)$, $\mathcal{W}(s,t)$, and $B(t)$ are uniformly bounded by M, and $\mathcal{V}(\cdot,t)$ and $\mathcal{W}(\cdot,t)$ are uniformly Lipschitz continuous with Lipschitz constant K. Formally, this last property means that, for any $s, s' \in [0,1]$ and $t \in [0,T]$,

$$\|\mathcal{V}(s',t) - \mathcal{V}(s,t)\|_F \leq K|s'-s| \quad \text{and} \quad \|\mathcal{W}(s',t) - \mathcal{W}(s,t)\|_F \leq K|s'-s|.$$

The boundedness and Lipschitz continuity of the weights are important features because they limit the statistical complexity of neural ODEs. More generally, norm-based bounds are a common approach in the statistical theory of deep learning (see, e.g., Bartlett et al., 2017, and references therein). Looking at the formula (3.7) for M and K, one can see in particular that the bounds diverge exponentially with T, providing an argument in favor of early stopping.

Our approach so far characterizes the large-depth limit of the neural network for a finite training time T, but two questions remain open. A first challenge is to characterize the value of the loss after training. A second one is to provide insight into the convergence of the optimization algorithm in the long-time limit, i.e., as T tends to infinity. To answer these questions, we move to the setting where the width of the network is large enough, which allows us to prove a Polyak-Łojasiewicz condition and thereby the long-time convergence of the training loss to zero.

3.4.2 Convergence in the long-time limit for wide networks

Proving convergence of gradient-based optimization algorithms for neural networks is a major difficulty in deep learning theory. One direction recently explored considers sufficiently wide neural networks, with the Polyak-Łojasiewicz (PL) condition. In our setting, they are written as follows (with the notation $Z^L = (V_k^L, W_k^L)_{k\in\{1,\dots,L\}}$):

Definition 3.5. *For $M, \mu > 0$, the residual network (3.3) is said to satisfy the (M,μ)-local PL condition around a set of parameters $(\bar{A}^L, \bar{Z}^L, \bar{B}^L)$ if, for every set of parameters (A^L, Z^L, B^L) such that*

$$\|A^L - \bar{A}^L\|_F \leqslant M, \qquad \sup_{k\in\{1,\dots,L\}} \|Z_k^L - \bar{Z}_k^L\|_F \leqslant M, \quad \|B^L - \bar{B}^L\|_F \leqslant M,$$

one has

$$\left\|\frac{\partial \ell^L}{\partial A^L}\right\|_F^2 + L\sum_{k=1}^{L}\left\|\frac{\partial \ell^L}{\partial Z_k^L}\right\|_F^2 + \left\|\frac{\partial \ell^L}{\partial B^L}\right\|_F^2 \geq \mu \ell^L,$$

where the loss ℓ^L is evaluated at the set of parameters (A^L, Z^L, B^L).

The next important point is to observe that, under the setup of Section 3.3 and some additional assumptions, the residual network satisfies the local PL condition of Definition 3.5.

Proposition 3.6. *Assume that the sample points (x_i, y_i) are i.i.d. such that $\|x_i\|_2 = \sqrt{q}$. Then there exist $c_1, \dots, c_4 > 0$ (depending only on σ) and $\delta > 0$ such that, if*

$$q \geq d + d', \quad m \geq c_1 n, \quad L \geq c_2 \sqrt{nq},$$

then, with probability at least $1 - \delta$, the residual network (3.3) satisfies the (M, μ)-local PL condition around its initialization, with $M = \dfrac{c_3}{\sqrt{nq}}$ and $\mu = \dfrac{c_4}{n\sqrt{nq}}$.

Proof. See Section 3.B.5. □

We emphasize that Proposition 3.6 requires the width m to scale only linearly with the sample size n, which improves on the literature (see Section 3.2). The other assumptions are mild. Note that our proof shows that the parameter δ is small if n grows at most polynomially with d (see Appendix 3.B.5).

We are now ready to state convergence in the long-time and large-depth limits to a global minimum of the empirical risk, when the local PL condition holds and the norm of the targets y_i is small enough.

Theorem 3.7. *Consider the residual network (3.3) with the training dynamics (3.4), and assume that the assumptions of Proposition 3.6 hold. Then there exist $C, \delta > 0$ such that, if $\frac{1}{n}\sum_{i=1}^n \|y_i\|^2 \leq C$, then, with probability as least $1 - \delta$, the gradient flow is well defined on $\mathbb{R}_+$, and, for $t \in \mathbb{R}_+$ and $L \in \mathbb{N}^*$,*

$$\ell^L(t) \leq \exp\Big(-\frac{C't}{n\sqrt{nq}}\Big)\ell^L(0), \tag{3.10}$$

*for some $C' > 0$ depending on σ. Moreover, the following statements hold **as** t **and** L **tend to infinity**:*

(i) There exist matrices $A_\infty \in \mathbb{R}^{q \times d}$ and $B_\infty \in \mathbb{R}^{d' \times q}$ such that $A^L(t)$ and $B^L(t)$ converge to A_∞ and B_∞.

(ii) There exists a Lipschitz continuous function $\mathcal{Z}_\infty : [0,1] \to \mathbb{R}^{q \times m} \times \mathbb{R}^{m \times q}$ such that

$$\mathcal{Z}^L : [0,1] \times \mathbb{R}_+ \to \mathbb{R}^{q \times m} \times \mathbb{R}^{m \times q}, \ (s,t) \mapsto \mathcal{Z}^L(s,t) = Z^L_{\lfloor (L-1)s \rfloor + 1}(t)$$

converges uniformly over $s \in [0,1]$ to $\mathcal{Z}_\infty = (\mathcal{V}_\infty, \mathcal{W}_\infty)$.

(iii) Uniformly over $s \in [0,1]$ and $x \in \mathcal{X}$, the hidden layer $h^L_{\lfloor Ls \rfloor}(t)$ converges to the solution at time s of the neural ODE

$$\begin{aligned} H(0) &= A_\infty x \\ \frac{dH}{ds}(s) &= \frac{1}{\sqrt{m}}\mathcal{V}_\infty(s)\sigma\Big(\frac{1}{\sqrt{q}}\mathcal{W}_\infty(s)H(s)\Big), \quad s \in [0,1]. \end{aligned}$$

(iv) Uniformly over $x \in \mathcal{X}$, the output $F^L(x;t)$ converges to $F_\infty(x) = B_\infty H(1)$. Furthermore, $F_\infty(x_i) = y_i$ for all $i \in \{1, \dots, n\}$.

Proof. See Section 3.B.6. □

This theorem proves two important results of separate interest. On the one hand, equation (3.10) shows the long-time convergence of the gradient flow for deep residual networks under the linear overparameterization assumption $m \geq c_1 n$ of Proposition 3.6. On the other hand, when both t and L tend to infinity, the network converges to a neural ODE that further interpolates the training data. Note that the order in which t and L tend to infinity does not matter by uniform convergence properties.

3.4.3 Generalizations to other architectures and initialization

To simplify the exposition, we have so far considered a particular residual architecture defined in (3.3). However, most of our results hold for a more general residual network of the form

$$h_{k+1}^L = h_k^L + \frac{1}{L} f(h_k^L, Z_{k+1}^L), \quad k \in \{0, \dots, L-1\}, \tag{3.11}$$

where $f : \mathbb{R}^q \times \mathbb{R}^p \to \mathbb{R}^q$ is a $\mathcal{C}^2$ function such that $f(0, \cdot) \equiv 0$ and $f(\cdot, z)$ is uniformly Lipschitz for z in any compact. All our results are shown in the appendix for this general model, except the PL condition of Proposition 3.6, which we prove only for the specific setup of Section 3.3. In particular, the conclusions of Theorem 3.4 hold for the general model (3.11), as well as those of Theorem 3.7 if the network satisfies a (M, μ)-local PL condition with μ sufficiently large (see Appendix 3.B for details).

It is easy to see that our residual network of interest (3.3) is a special case of the general model (3.11) if σ satisfies the assumptions of Section 3.3. However, other choices are possible, such as convolutional layers or a Lipschitz continuous version of Transformer (Kim et al., 2021). This latter application is particularly interesting in the light of the literature analyzing the Transformer architecture from a neural ODE point of view (Lu et al., 2019; Sander et al., 2022a; Geshkovski et al., 2023).

Moreover, the initialization assumption made in Section 3.3 can also be relaxed to include any so-called *smooth* initialization of the weights (see Marion et al. (2022)). A smooth initialization corresponds to taking $V_k^L(0)$ and $W_k^L(0)$ as discretizations of some Lipschitz continuous functions $\mathcal{V}_0 : [0,1] \to \mathbb{R}^{q \times m}$ and $\mathcal{W}_0 : [0,1] \to \mathbb{R}^{m \times q}$, that is, for $k \in \{1, \dots, L\}$, $V_k^L(0) = \mathcal{V}_0(\frac{k}{L})$ and $W_k^L(0) = \mathcal{W}_0(\frac{k}{L})$. A typical concrete example is to let the entries of $\mathcal{V}_0$ and $\mathcal{W}_0$ be independent Gaussian processes with expectation zero and squared exponential covariance $K(x, x') = \exp(-\frac{(x-x')^2}{2\ell^2})$, for some $\ell > 0$. As shown by Proposition 3.2, a smooth initialization means that the network discretizes a neural ODE.

3.5 Numerical experiments

We now present numerical experiments to validate our theoretical findings, using both synthetic and real-world data. Experimental details are given in Appendix 3.E. Our code will be open sourced.

3.5.1 Synthetic data

We consider the residual network (3.3) with the initialization scheme of Section 3.3. So, the V_k^L are initialized to zero and the W_k^L to weight-tied standard Gaussian matrices. To ease the presentation, we consider the case where $q = d = d'$, and we do not train the weights A^L and B^L, which therefore stay equal to the identity. The activation function is GELU (Hendrycks and Gimpel, 2016), which is a smooth approximation of ReLU: $x \mapsto \max(x, 0)$. The sample points

$(x_i, y_i)_{1\leq i\leq n}$ follow independent standard Gaussian distributions. Note that it does not hurt to take x and y independent since, in this subsection, our focus is on optimization results only and not on statistical aspects. The mean-squared error is minimized using full-batch gradient descent. The following experiments exemplify the large-depth ($t \in [0, T]$, $L \to \infty$) and long-time ($t \to \infty$, L finite) limits.

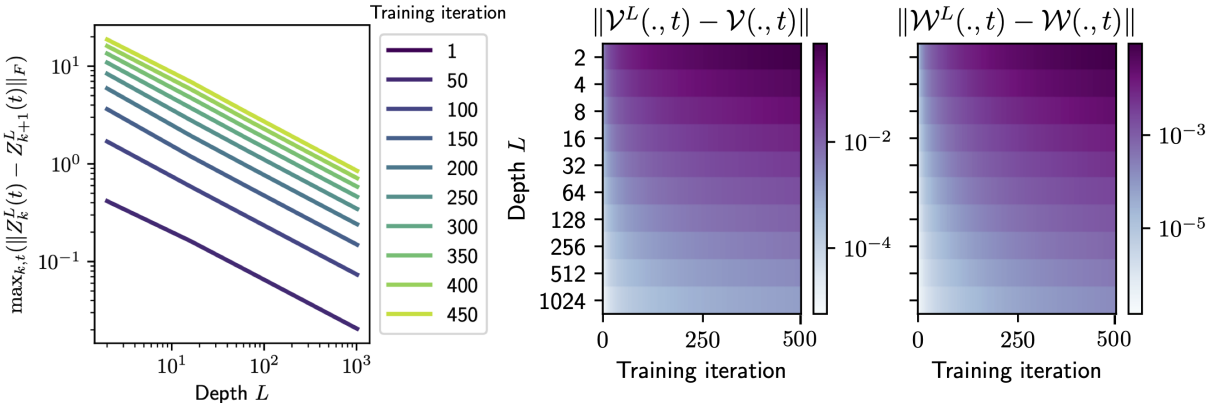

Figure 3.1: **Left**: $1/L$ convergence of the maximum distance between two successive weight matrices $\max_{1\leq k\leq L, t\in[0,T]}(\|Z_k^L(t) - Z_{k+1}^L(t)\|_F)$. **Right**: uniform convergence of $\mathcal{Z}^L$ to its large-depth limit $\mathcal{Z}$. Here, for a matrix-valued function f, $\|f\|$ denotes $(\int_0^1 \|f(s)\|_F^2 ds)^{1/2}$.

Large-depth limit. We illustrate key insights of Proposition 3.3 and Theorem 3.4, with $T = 500$. In Figure 3.1 (left), we plot the maximum distance between two successive weight matrices, i.e., $\max_{1\leq k\leq L, t\in[0,T]}(\|Z_k^L(t) - Z_{k+1}^L(t)\|_F)$, for different values of L and training time $t \in [0, T]$. We observe a $1/L$ convergence rate, as predicted by Proposition 3.3. Moreover, for a fixed L, the distance between two successive weight matrices increases with the training time, however at a much slower pace than the exponential upper bound on K given in identity (3.7). Figure 3.1 (right) depicts the uniform convergence of $\mathcal{Z}^L$ to its large-depth limit $\mathcal{Z}$, illustrating statement (ii) of Theorem 3.4. The function $\mathcal{Z}$ is computed using $\mathcal{Z}^L$ for $L = 2^{14}$. Note that the convergence is slower for larger training times.

Long-time limit. We now turn to the long-time training setup, training for 80,000 iterations with $L = 64$. In Figure 3.2, we plot a specific (randomly-chosen) entry of matrices V_k^L and W_k^L across layers, for different training times. This illustrates Theorem 3.7 in a practical setting since, visually, the weights behave as a Lipschitz continuous function for any training time and converge to a Lipschitz continuous function as $t \to \infty$. We also display the loss as a function of the training time, corroborating the convergence of the loss to zero in Theorem 3.7.

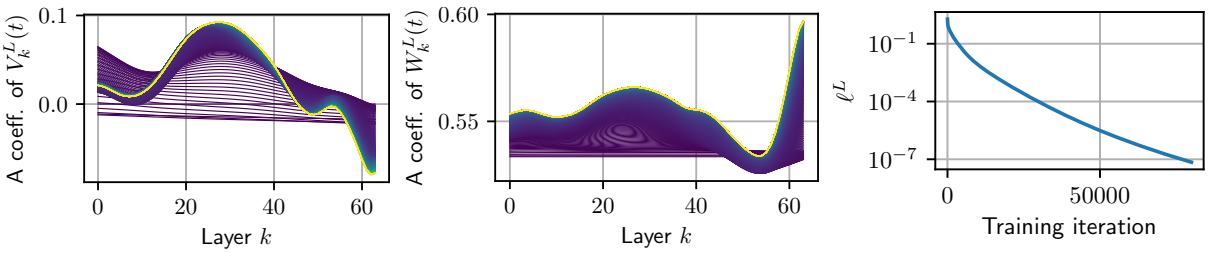

Figure 3.2: **Left**: Randomly-chosen entry of the weight matrices across layers (x-axis) for various training times t (lighter color indicates higher training time). **Right**: Loss against training time.

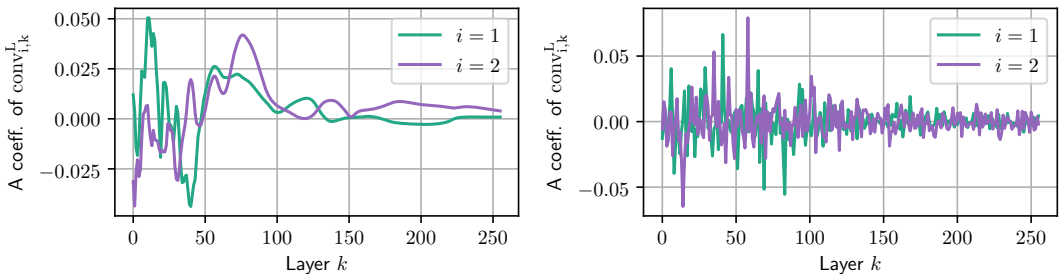

Figure 3.3: Random entries of the convolutions across layers (x-axis) after training. **Left:** Weight-tied initialization leads to smooth weights. **Right:** i.i.d. initialization leads to non-smooth weights.

3.5.2 Real-world data

We now investigate the properties of deep residual networks on the CIFAR 10 dataset (Krizhevsky, 2009). We deviate from the mathematical model (3.3) by using convolutions instead of fully connected layers. More precisely, A^L is replaced by a trainable convolutional layer, and the residual layers write

$$h^L_{k+1} = h^L_k + \frac{1}{L}\mathrm{bn}^L_{2,k}(\mathrm{conv}^L_{2,k}(\sigma(\mathrm{bn}^L_{1,k}(\mathrm{conv}^L_{1,k}(h^L_k))))), \quad k \in \{0, \dots, L-1\},$$

where $\mathrm{conv}^L_{i,k}$ are convolutions and $\mathrm{bn}^L_{i,k}$ are batch normalizations. The output of the residual layers is mapped to logits through a linear layer B^L. We initialize $\mathrm{bn}^{\mathrm{L}}_{2,\mathrm{k}}$ to 0, and $\mathrm{bn}^{\mathrm{L}}_{1,\mathrm{k}}$ and $\mathrm{conv}^{\mathrm{L}}_{\mathrm{i,k}}$ either to weight-tied or to i.i.d. Gaussian. Table 3.1 reports the accuracy of the trained network, and whether it has Lipschitz continuous (or smooth) weights after training, depending on the activation function σ and on the initialization scheme. To assess the smoothness of the weights, we simply resort to visual inspection. For example, Figure 3.3 (left) shows two random entries of the convolutions across layers with GELU and a weight-tied initialization: the smoothness is preserved after training. Smooth weights indicate that the residual network discretizes a neural ODE (see, e.g., Proposition 3.2). On the contrary, if an i.i.d. initialization is used, smoothness is not preserved after training, as shown in Figure 3.3 (right), and the residual network does not discretize a neural ODE.

Act. function	Init. scheme	Train Acc.	Test Acc.	Smooth trained weights
Identity	Weight-tied	56.5 ± 0.1	59.8 ± 0.7	✓
	i.i.d.	56.1 ± 0.3	59.6 ± 0.7	×
GELU	Weight-tied	80.5 ± 0.7	79.9 ± 0.2	✓
	i.i.d.	89.8 ± 0.5	85.7 ± 0.1	×
ReLU	Weight-tied	97.4 ± 0.6	88.1 ± 0.1	×
	i.i.d.	98.4 ± 0.1	88.4 ± 0.5	×

Table 3.1: Accuracy and smoothness of the trained weights depending on the choice of activation function σ and initialization scheme. We display the median over 5 runs and the interquartile range between the first and third quantile. Smooth weights correspond to a neural ODE structure.

Table 3.1 conveys several important messages. First, in accordance with our theory (Theorem 3.4), we obtain a neural ODE structure when using a smooth activation function and weight-tied

initialization (lines 1 and 3 of Table 3.1). This is not the case when using the non-smooth ReLU activation and/or i.i.d. initialization. In fact, we prove in Appendix 3.D that the smoothness of the weights is lost when training with ReLU in a simple setting, confirming this experimental observation. Furthermore, the third line of Table 3.1 shows that it is possible to obtain a reasonable accuracy with a neural ODE structure, which, as emphasized in Section 3.1, also comes with theoretical and practical advantages. Nevertheless, we obtain an improvement in accuracy in the cases corresponding to non-smooth weights, i.e., to a residual network that does *not* discretize an ODE. Extending our theory to such cases is left for future work.

3.6 Conclusion

We study the convergence of deep residual networks to neural ODEs. When properly scaled and initialized, residual networks trained with fixed-horizon gradient flow converge to neural ODEs as the depth tends to infinity. This result holds for very general architectures. In the case where both training time and depth tend to infinity, convergence holds under a local Polyak-Łojasiewicz condition. We prove such a condition for a family of deep residual networks with linear overparameterization.

The setting of neural ODE-like networks comes with strong guarantees, at the cost of some performance gap when compared with i.i.d. initialization as highlighted by the experimental section. Extending the mathematical large-depth study to the i.i.d. case is an interesting problem for future research. Previous work suggests that the correct limit object is then a stochastic differential equation instead of an ODE, such as Cohen et al. (2021); Cont et al. (2022); Marion et al. (2022).

3.A Some results for general residual networks

Lipschitz continuity. Let $(\mathscr{U}, \|\cdot\|)$, $(\mathscr{V}, \|\cdot\|)$, and $(\mathscr{W}, \|\cdot\|)$ be generic normed spaces. Then a function of two variables $g : \mathscr{U} \times \mathscr{V} \to \mathscr{W}$ is:

(*i*) (Globally) Lipschitz continuous if there exists $K \geq 0$ such that, for $(u, v), (u', v') \in \mathscr{U} \times \mathscr{V}$,

$$\|g(u, v) - g(u', v)\| \leq K\|u - u'\| + K\|v - v'\|.$$

(*ii*) Locally Lipschitz continuous in its first variable if, for any compacts $E \subset \mathscr{U}, E' \subset \mathscr{V}$, there exists $K \geq 0$ such that, for $(u, v), (u', v) \in E \times E'$,

$$\|g(u, v) - g(u', v)\| \leq K\|u - u'\|.$$

Equivalent definitions hold for a function of one variable. Moreover, $g(\cdot, v)$ is said to be uniformly Lipschitz continuous for v in $\mathscr{V}$ if there exists $K \geq 0$ such that, for $(u, v), (u', v) \in \mathscr{U} \times \mathscr{V}$,

$$\|g(u, v) - g(u', v)\| \leq K\|u - u'\|,$$

and uniformly Lipschitz continuous for v in any compact if, for any compact $E' \subset \mathscr{V}$, there exists $K \geq 0$ such that, for $(u, v), (u', v) \in \mathcal{U} \times E'$,

$$\|g(u, v) - g(u', v)\| \leq K\|u - u'\|.$$

Throughout, we refer to a Lipschitz continuous function with Lipschitz constant $K \geq 0$ as K-Lipschitz.

Model. As explained in Section 3.4.3, most of our results are proven for the general residual network

$$\begin{aligned} h_0^L(t) &= A^L(t)x \\ h_{k+1}^L(t) &= h_k^L(t) + \frac{1}{L} f(h_k^L(t), Z_{k+1}^L(t)), \quad k \in \{0, \dots, L-1\}, \\ F^L(x;t) &= B^L(t) h_L^L(t), \end{aligned} \tag{3.12}$$

where $Z^L(t) = (Z_1^L(t), \dots, Z_L^L(t)) \in (\mathbb{R}^p)^L$ and $f : \mathbb{R}^q \times \mathbb{R}^p \to \mathbb{R}^q$ is a $\mathcal{C}^2$ function such that $f(0, \cdot) \equiv 0$ and $f(\cdot, z)$ is uniformly Lipschitz for z in any compact. Let us introduce the backpropagation equations, which are instrumental in the study of the gradient flow dynamics. These equations define the backward state $p_k^L(t) \in \mathbb{R}^q$ through the backward recurrence

$$\begin{aligned} p_L^L(t) &= 2B^L(t)^\top (F^L(x;t) - y) \\ p_k^L(t) &= p_{k+1}^L(t) + \frac{1}{L} \partial_1 f(h_k^L(t), Z_{k+1}^L(t)) p_{k+1}^L(t), \quad k \in \{0, \dots, L-1\}, \end{aligned} \tag{3.13}$$

where $\partial_1 f \in \mathbb{R}^{q \times q}$ stands for the Jacobian matrix of f with respect to its first argument. Similarly, we let $\partial_2 f \in \mathbb{R}^{q \times p}$ be the Jacobian matrix of f with respect to its second argument. For a sample $(x_i, y_i)_{1 \le i \le n} \in (\mathcal{X} \times \mathcal{Y})^n$, we let $h_{k,i}^L(t)$ and $p_{k,i}^L(t)$ be, respectively, the hidden layer $h_k^L(t)$ and the backward state $p_k^L(t)$ associated with the i-th input x_i. Denoting the mean squared error associated with the sample by ℓ^L, we have, by the chain rule,

$$\frac{\partial \ell^L}{\partial A^L}(t) = \frac{1}{n} \sum_{i=1}^n p_{0,i}^L(t) x_i^\top \tag{3.14}$$

$$\frac{\partial \ell^L}{\partial Z_k^L}(t) = \frac{1}{nL} \sum_{i=1}^n \partial_2 f(h_{k-1,i}^L(t), Z_k^L(t))^\top p_{k-1,i}^L(t), \quad k \in \{1, \dots, L\}, \tag{3.15}$$

$$\frac{\partial \ell^L}{\partial B^L}(t) = \frac{2}{n} \sum_{i=1}^n (F^L(x_i;t) - y_i) h_{L,i}^L(t)^\top. \tag{3.16}$$

Initialization. The parameters $(Z_k^L(t))_{1 \le k \le L}$ are initialized to $Z_k^L(0) = Z^{\text{init}}\left(\frac{k}{L}\right)$, where $Z^{\text{init}} : [0,1] \to \mathbb{R}^p$ is a Lipschitz continuous function. Furthermore, we initialize $A^L(0)$ to some matrix $A^{\text{init}} \in \mathbb{R}^{q \times d}$ and $B^L(0) = B^{\text{init}} \in \mathbb{R}^{d' \times q}$. Note that this initialization scheme is a generalization of the one presented in Section 3.3.

Additional notation. For a vector x, $\|x\|$ denotes the Euclidean norm. For a matrix A, the operator norm induced by the Euclidean norm is denoted by $\|A\|_2$, and the Frobenius norm is denoted by $\|A\|_F$. Finally, we use the notation A^L (resp. Z_k^L, B^L) to denote the function $t \mapsto A^L(t)$ (resp. $t \mapsto Z_k^L(t)$, $t \mapsto B^L(t)$), since the parameters are considered as functions of the training time throughout this appendix.

Overview of Appendix A. First, in Section 3.A.1, we study the case of the (clipped) gradient flow (3.5). We show that the weights and the difference between successive weights are bounded during the entire training. Section 3.A.2 shows a similar result for the standard gradient flow (3.4) under a PL condition. In Section 3.A.3, we show a generalized version of the Arzelà-Ascoli theorem, which allows us to prove the existence of a converging subsequence of the weights in the large-depth limit. Section 3.A.4 is devoted to the convergence of the Euler scheme for parameterized ODEs. We then proceed to prove in Section 3.A.5 our main result, i.e., the

large-depth convergence of the gradient flow. The key step is to establish the uniqueness of the adherence point of the weights. Finally, in Section 3.A.6, we prove the existence of a double limit for the weights and the hidden states when both the depth and the training time tend to infinity.

3.A.1 The trained weights are bounded in the finite training-time setup

Before stating the result, let us introduce the notation $\partial_{22} f(h,z) \in \mathbb{R}^{q\times p\times p}$, which is the third-order tensor of second partial derivatives of f with respect to z. We endow the space $\mathbb{R}^{q\times p\times p}$ with the operator norm $\|\cdot\|_2$ induced by the Euclidean norm in $\mathbb{R}^p$ and the $\|\cdot\|_2$ norm in $\mathbb{R}^{q\times p}$. In other words,

$$\|\partial_{22} f(h,z)\|_2 = \sup_{u\in\mathbb{R}^p, \|u\|=1} \|\partial_{22} f(h,z)u\|_2,$$

where $\partial_{22} f(h,z)u \in \mathbb{R}^{q\times p}$ is the tensor product of $\partial_{22} f(h,z)$ against u. Similarly, $\partial_{21} f(h,z) \in \mathbb{R}^{q\times p\times q}$ denotes the third-order tensor of cross second partial derivatives of f, and the space $\mathbb{R}^{q\times p\times q}$ is endowed with the operator norm $\|\cdot\|_2$ induced by the Euclidean norm in $\mathbb{R}^q$ and the $\|\cdot\|_2$ norm in $\mathbb{R}^{q\times p}$.

Proposition 3.8. *Consider the residual network (3.12) initialized as explained in Appendix 3.A and trained with the gradient flow (3.5) on $[0,T]$, for some $T \in (0,\infty)$. Let*

$$M_\pi = \max\Big(\max_{A\in\mathbb{R}^{q\times d}} \|\pi(A)\|_F, \max_{Z\in\mathbb{R}^p} \|\pi(Z)\|, \max_{B\in\mathbb{R}^{d'\times q}} \|\pi(B)\|_F \Big),$$
$$M_0 = \max\Big(\|A^{\text{init}}\|_F, \sup_{s\in[0,1]} \|Z^{\text{init}}(s)\|, \|B^{\text{init}}\|_F \Big) \quad \text{and} \quad M = M_0 + TM_\pi.$$

Then the gradient flow is well defined on $[0,T]$, and, for $t\in[0,T]$, $L\in\mathbb{N}^*$, and $k\in\{1,\dots,L\}$,

$$\|A^L(t)\|_F \le M, \quad \|Z_k^L(t)\| \le M, \quad \text{and} \quad \|B^L(t)\|_F \le M. \tag{3.17}$$

Moreover, there exist $\alpha,\beta>0$ such that, for $t\in[0,T]$ and $k\in\{1,\dots,L-1\}$,

$$\|Z_{k+1}^L(t) - Z_k^L(t)\| \le \Big(\|Z_{k+1}^L(0) - Z_k^L(0)\| + \frac{\beta T}{L} \Big) e^{\alpha T}.$$

The following expressions for α and β hold:

$$\alpha = 2e^K K' M(e^K M^2 M_X + M_Y) \quad \text{and} \quad \beta = 2Ke^K M(K + e^K K' M M_X)(e^K M^2 M_X + M_Y),$$

where

$$M_X = \sup_{x\in\mathcal{X}} \|x\|, \quad M_Y = \sup_{y\in\mathcal{Y}} \|y\|, \quad K_1 = \sup_{\|z\|\le M} \big\|\partial_1 f(h,z)\big\|_2 \tag{3.18}$$
$$E = \{(h,z)\in\mathbb{R}^d\times\mathbb{R}^p, \|h\| \le e^{K_1} M M_X,\ \|z\|\le M\} \tag{3.19}$$
$$K_2 = \sup_{(h,z)\in E} \big\|\partial_2 f(h,z)\big\|_2, \quad K = \max(K_1,K_2)$$
$$K' = \sup_{(h,z)\in E} \Big(\max\big(\big\|\partial_{22} f(h,z)\big\|_2, \big\|\partial_{21} f(h,z)\big\|_2 \big) \Big).$$

Proof. The time-independent dynamics

$$(A^L, Z_k^L, B^L) \mapsto \Big(\pi\Big(-\frac{\partial \ell^L}{\partial A^L} \Big), \pi\Big(-L\frac{\partial \ell^L}{\partial Z_k^L} \Big), \pi\Big(-\frac{\partial \ell^L}{\partial B^L} \Big) \Big)$$

defining the gradient flow (3.5) are locally Lipschitz continuous, hence the gradient flow is defined on a maximal interval $[0, T_{\max})$ by the Picard-Lindelöf theorem (see Lemma 3.19). Let us show by contradiction that $T_{\max} = T$. Assume that $T_{\max} < T$. If this is true, again by the Picard-Lindelöf theorem, we know that the parameters diverge to infinity at $T_{\max}$. However, for any $t \in [0, T_{\max})$, we have

$$\|A^L(t)\|_F \le \|A^L(0)\|_F + \int_0^t \Big\|\frac{dA^L}{dt}(\tau)\Big\|_F d\tau \le M_0 + \int_0^t M_\pi d\tau \le M_0 + TM_\pi = M.$$

Bounds on B^L and Z_k^L by M can be shown similarly. This contradicts the divergence of the parameters at $t = T_{\max}$. We conclude that the gradient flow is well defined on $[0, T]$ and that the bounds (3.17) hold.

It remains to bound the difference $\|Z_{k+1}^L(t) - Z_k^L(t)\|$. We have, for $t \in [0, T]$ and $k \in \{1, \dots, L-1\}$,

$$\begin{aligned}
\Big\|\frac{dZ_{k+1}^L}{dt}(t) - \frac{dZ_k^L}{dt}(t)\Big\| &= L\Big\|\frac{\partial \ell^L}{\partial Z_{k+1}^L}(t) - \frac{\partial \ell^L}{\partial Z_k^L}(t)\Big\| \\
&\le \sum_{i=1}^n \frac{1}{n}\big\|\partial_2 f(h_{k,i}^L(t), Z_{k+1}^L(t))^\top p_{k,i}^L(t) - \partial_2 f(h_{k-1,i}^L(t), Z_k^L(t))^\top p_{k-1,i}^L(t)\big\| \\
&\le \frac{1}{n}\sum_{i=1}^n \big\|\partial_2 f(h_{k,i}^L(t), Z_{k+1}^L(t))\big\|_2 \big\|p_{k,i}^L(t) - p_{k-1,i}^L(t)\big\| \\
&\quad + \big\|p_{k-1,i}^L(t)\big\| \big\|\partial_2 f(h_{k,i}^L(t), Z_{k+1}^L(t)) - \partial_2 f(h_{k-1,i}^L(t), Z_k^L(t))\big\|_2
\end{aligned} \tag{3.20}$$

Furthermore, for $t \in [0, T]$, $k \in \{0, \dots, L-1\}$, and $i \in \{1, \dots, n\}$,

$$\|h_{k+1,i}^L(t)\| = \|h_{k,i}^L(t) + \frac{1}{L} f(h_{k,i}^L(t), Z_{k+1}^L(t))\| \le (1 + \frac{K_1}{L})\|h_{k,i}^L(t)\|,$$

since $f(\cdot, Z_{k+1}^L(t))$ is K_1-Lipschitz, where K_1 is defined by (3.18), and $f(0, Z_{k+1}^L(t)) = 0$. Therefore, for any $k \in \{1, \dots, L\}$,

$$\|h_{k,i}^L(t)\| \le e^{K_1}\|h_{0,i}^L(t)\| = e^{K_1}\|A^L(t)x_i\| \le e^{K_1} M M_X. \tag{3.21}$$

This bound shows that the pair $(h_{k,i}^L(t), Z_{k+1}^L(t))$ belongs to the compact E defined in (3.19) for every $t \in [0, T]$, $k \in \{1, \dots, L\}$, and $i \in \{1, \dots, n\}$. In particular, $\|\partial_2 f(h_{k-1,i}^L(t), Z_k^L(t))\|_2 \le K$, and

$$\begin{aligned}
&\big\|\partial_2 f(h_{k,i}^L(t), Z_{k+1}^L(t)) - \partial_2 f(h_{k-1,i}^L(t), Z_k^L(t))\big\|_2 \\
&\qquad \le K'\|h_{k,i}^L(t) - h_{k-1,i}^L(t)\| + K'\|Z_{k+1}^L(t) - Z_k^L(t)\|.
\end{aligned}$$

Returning to (3.20), we obtain

$$\begin{aligned}
\Big\|\frac{dZ_{k+1}^L}{dt}(t) - \frac{dZ_k^L}{dt}(t)\Big\| &\le \frac{1}{n}\sum_{i=1}^n K\|p_{k,i}^L(t) - p_{k-1,i}^L(t)\| \\
&\quad + K'\|p_{k-1,i}^L(t)\|\big(\|h_{k,i}^L(t) - h_{k-1,i}^L(t)\| + \|Z_{k+1}^L(t) - Z_k^L(t)\|\big).
\end{aligned}$$

For $k \in \{1, \dots, L\}$ and $i \in \{1, \dots, n\}$,

$$\big\|p_{k,i}^L(t) - p_{k-1,i}^L(t)\big\| = \frac{1}{L}\big\|\partial_1 f(h_{k-1,i}^L(t), Z_k^L(t)) p_{k,i}^L(t)\big\| \le \frac{K}{L}\big\|p_{k,i}^L(t)\big\|,$$

and, similarly,

$$\big\|h^L_{k,i}(t) - h^L_{k-1,i}(t)\big\| = \frac{1}{L}\|f(h^L_{k-1,i}(t), Z^L_k(t))\| \leq \frac{K}{L}\big\|h^L_{k-1,i}(t)\big\| \leq \frac{Ke^K M M_X}{L}.$$

Thus,

$$\Big\|\frac{dZ^L_{k+1}}{dt}(t) - \frac{dZ^L_k}{dt}(t)\Big\| \leq \frac{1}{n}\sum_{i=1}^n \|p^L_{k,i}(t)\|\Big(\frac{K^2}{L} + \frac{K'K}{L}e^K M M_X + K'\|Z^L_{k+1}(t) - Z^L_k(t)\|\Big).$$

Moreover, for $k \in \{0, \dots, L\}$ and $i \in \{1, \dots, n\}$,

$$\|p^L_{k,i}(t)\| \leq \|p^L_{k+1,i}(t)\| + \frac{1}{L}\big\|\partial_1 f(h^L_{k,i}(t), Z^L_{k+1}(t))p^L_{k+1,i}(t)\big\| \leq \|p^L_{k+1,i}(t)\| + \frac{K}{L}\|p^L_{k+1,i}(t)\|.$$

Hence

$$\begin{aligned}\|p^L_{k,i}(t)\| &\leq e^K\|p^L_{L,i}(t)\| = 2e^K\|B^L(t)^\top(F^L(x_i;t) - y_i)\| \\ &\leq 2e^K M\big(\|B^L(t)h^L_{L,i}(t)\| + \|y_i\|\big) \leq 2e^K M(e^K M^2 M_X + M_Y),\end{aligned}$$

where we use (3.17) and (3.21) for the last inequality. Putting all the pieces together, we obtain

$$\Big\|\frac{dZ^L_k}{dt}(t) - \frac{dZ^L_{k+1}}{dt}(t)\Big\| \leq \alpha\|Z^L_k(t) - Z^L_{k+1}(t)\| + \frac{\beta}{L}.$$

Integrating between 0 and t, we see that

$$\|Z^L_{k+1}(t) - Z^L_k(t)\| \leq \|Z^L_{k+1}(0) - Z^L_k(0)\| + \frac{\beta t}{L} + \int_0^t \alpha\|Z^L_k(\tau) - Z^L_{k+1}(\tau)\|d\tau.$$

Applying Grönwall's inequality (see, e.g., Dragomir, 2003), we conclude that $\|Z^L_{k+1}(t) - Z^L_k(t)\| \leq (\|Z^L_{k+1}(0) - Z^L_k(0)\| + \frac{\beta T}{L})e^{\alpha T}$, as desired. □

3.A.2 The trained weights are bounded under the local PL condition

Proposition 3.9. *Consider the residual network (3.12) initialized as explained in Appendix 3.A and trained with the gradient flow (3.4) on $[0, \infty]$. Then, for $M > 0$, there exists $\mu > 0$ such that, if the residual network satisfies the (M, μ)-local PL condition (3.5) around its initialization for any $L \in \mathbb{N}^*$, then:*

(i) The gradient flow is well defined on $\mathbb{R}_+$, and, for $t \in \mathbb{R}_+$, $L \in \mathbb{N}^*$, and $k \in \{1, \dots, L\}$,

$$\|A^L(t)\|_F \leq M_A, \quad \|Z^L_k(t)\| \leq M_Z, \quad \text{and} \quad \|B^L(t)\|_F \leq M_B,$$

where

$$M_A = \|A^{\text{init}}\|_2 + M, \quad M_Z = \sup_{s\in[0,1]}\|Z^{\text{init}}(s)\| + M, \quad \text{and} \quad M_B = \|B^{\text{init}}\|_2 + M.$$

(ii) There exists $\tilde{K} > 0$ such that, for $t \in \mathbb{R}_+$, $L \in \mathbb{N}^*$, and $k \in \{1, \dots, L\}$,

$$\|Z^L_k(t) - Z^L_{k+1}(t)\| \leq \frac{\tilde{K}}{L}.$$

(iii) There exists a bounded integrable function $b : \mathbb{R}_+ \to \mathbb{R}$ such that, for $t \in \mathbb{R}_+$, $L \in \mathbb{N}^*$, and $k \in \{1, \dots, L\}$,

$$\max\Big(\Big\|\frac{dA^L}{dt}(t)\Big\|, \Big\|\frac{dZ_k^L}{dt}(t)\Big\|, \Big\|\frac{dB^L}{dt}(t)\Big\|\Big) \le b(t)$$

(iv) $A^L(t)$, $B^L(t)$, and $Z_k^L(t)$ admit a limit uniformly over $L \in \mathbb{N}^*$ and $k \in \{1, \dots, L\}$ as $t \to \infty$.

(v) For $t \in \mathbb{R}_+$ and $L \in \mathbb{N}^*$, $\ell^L(t) \le e^{-\mu t}\ell^L(0)$.

Moreover, the following expression for μ hold:

$$\mu = \max(M_B K, M_B M_X, M_A M_X)\frac{8e^K}{M}\sup_{L\in\mathbb{N}^*}\sqrt{\ell^L(0)}, \tag{3.22}$$

where

$$\begin{aligned} M_X &= \sup_{x\in\mathcal{X}}\|x\|, \quad K_1 = \sup_{\|z\|\le M_Z}\big\|\partial_1 f(h,z)\big\| \\ E &= \{(h,z) \in \mathbb{R}^d \times \mathbb{R}^p, \|h\| \le e^{K_1} M_A M_X,\ \|z\| \le M_Z\} \\ K_2 &= \sup_{(h,z)\in E}\big\|\partial_2 f(h,z)\big\|, \quad K = \max(K_1, K_2). \end{aligned}$$

Proof. Let $M > 0$, μ defined by (3.22), and assume that the residual network satisfies the (M,μ)-local PL condition (3.5) around its initialization for any $L \in \mathbb{N}^*$.

The time-independent dynamics

$$(A^L, Z_k^L, B^L) \mapsto \Big(-\frac{\partial \ell^L}{\partial A^L}, -L\frac{\partial \ell^L}{\partial Z_k^L}, -\frac{\partial \ell^L}{\partial B^L}\Big)$$

defining the gradient flow (3.5) are locally Lipschitz continuous, hence the gradient flow is defined on a maximal interval $[0, T_{\max})$ by the Picard-Lindelöf theorem (see Lemma 3.19). Let us show by contradiction that $T_{\max} = \infty$. Assume that $T_{\max} < \infty$. If this is true, again by the Picard-Lindelöf theorem, we know that the parameters diverge to infinity at $T_{\max}$. In particular, there exist $t \in (0, T_{\max})$ and $k \in \{1, \dots, L\}$ such that

$$\|A^L(t) - A^L(0)\|_F > M \text{ or } \|Z_k^L(t) - Z_k^L(0)\| > M \text{ or } \|B^L(t) - B^L(0)\|_F > M.$$

Let $t^* \in (0, T_{\max})$ be the infimum of such times t. Then, for $t < t^*$ and $k \in \{1, \dots, L\}$,

$$\|A^L(t) - A^L(0)\|_F \le M \text{ and } \|Z_k^L(t) - Z_k^L(0)\| \le M \text{ and } \|B^L(t) - B^L(0)\|_F \le M, \tag{3.23}$$

and, by continuity of A^L, B^L, and Z_k^L, these inequalities also hold for $t = t^*$. By definition, this means that the (M,μ)-local PL condition is satisfied for $t \le t^*$, and ensures that

$$\Big\|\frac{\partial \ell^L}{\partial A^L}(t)\Big\|_F^2 + L\sum_{k=1}^L\Big\|\frac{\partial \ell^L}{\partial Z_k^L}(t)\Big\|^2 + \Big\|\frac{\partial \ell^L}{\partial B^L}(t)\Big\|_F^2 \ge \mu\ell^L(t).$$

Therefore, by definition of the gradient flow (3.4),

$$\begin{aligned} \frac{d\ell^L}{dt}(t) &= \Big\langle\frac{\partial \ell^L}{\partial A^L}(t), \frac{dA^L}{dt}(t)\Big\rangle + \sum_{k=1}^L\Big\langle\frac{\partial \ell^L}{\partial Z_k^L}(t), \frac{dZ_k^L}{dt}(t)\Big\rangle + \Big\langle\frac{\partial \ell^L}{\partial B^L}(t), \frac{dB^L}{dt}(t)\Big\rangle \\ &= -\Big\|\frac{\partial \ell^L}{\partial A^L}(t)\Big\|_F^2 - L\sum_{k=1}^L\Big\|\frac{\partial \ell^L}{\partial Z_k^L}(t)\Big\|^2 - \Big\|\frac{\partial \ell^L}{\partial B^L}(t)\Big\|_F^2 \\ &\le -\mu\ell^L(t). \end{aligned}$$

Thus, by Grönwall's inequality, for $t \le t^*$,

$$\ell^L(t) \le e^{-\mu t}\ell^L(0). \tag{3.24}$$

Furthermore, by (3.23) and the definition of M_A, M_B, M_Z, we have, for $t \le t^*$ and $k \in \{1, \dots, L\}$,

$$\|A^L(t)\|_F \le M_A, \quad \|Z_k^L(t)\| \le M_Z, \quad \text{and} \quad \|B^L(t)\|_F \le M_B.$$

A quick scan through the proof of Proposition 3.8 reveals that by similar arguments, we have, for $t \le t^*$, $k \in \{1, \dots, L\}$, and $i \in \{1, \dots, n\}$,

$$(h_{k-1,i}^L(t), Z_k^L(t)) \in E \quad \text{and} \quad \|p_{k-1,i}^L(t)\| \le 2e^K\|p_{L,i}^L(t)\| \le 2e^K M_B\|F^L(x_i;t) - y_i\|.$$

Thus, for $k \in \{0, \dots, L\}$,

$$\frac{1}{n}\sum_{i=1}^n \|p_{k,i}^L(t)\| \le \frac{2e^K M_B}{n}\sum_{i=1}^n \|F^L(x_i;t) - y_i\| \le 2e^K M_B\sqrt{\ell^L(t)} \le 2e^K M_B e^{-\frac{\mu t}{2}}\sqrt{\ell^L(0)}, \tag{3.25}$$

where the second inequality is a consequence of the Cauchy-Schwartz inequality. Let us now bound $\|Z_k^L(t^*) - Z_k^L(0)\|$. We have, for $k \in \{1, \dots, L\}$,

$$\begin{aligned}
\|Z_k^L(t^*) - Z_k^L(0)\| &\le \int_0^{t^*} \Big\|\frac{dZ_k^L}{dt}(t)\Big\| dt \\
&\le \frac{1}{n}\sum_{i=1}^n \int_0^{t^*} \big\|\partial_2 f(h_{k-1,i}^L(t), Z_k^L(t))^\top p_{k-1,i}^L(t)\big\| dt \\
&\qquad \text{(by (3.15))}. \\
&\le \frac{K}{n}\sum_{i=1}^n \int_0^{t^*} \|p_{k-1,i}^L(t)\| dt,
\end{aligned}$$

since $(h_{k-1,i}^L(t), Z_k^L(t)) \in E$ and $\|\partial_2 f(h,z)\| \le K$ for $(h,z) \in E$. Therefore, by (3.25),

$$\|Z_k^L(t^*) - Z_k^L(0)\| \le 2Ke^K M_B \int_0^{t^*} e^{-\frac{\mu t}{2}}\sqrt{\ell^L(0)} dt \le \frac{4Ke^K M_B}{\mu}\sqrt{\ell^L(0)} \le \frac{M}{2},$$

where the last inequality is a consequence of the definition of μ. Similarly, by (3.14) and (3.25),

$$\begin{aligned}
\|A^L(t^*) - A^L(0)\|_F &\le \int_0^{t^*} \Big\|\frac{dA^L}{dt}(t)\Big\|_F dt \\
&\le \int_0^{t^*} \frac{1}{n}\sum_{i=1}^n \big\|p_{0,i}^L(t)x_i^\top\big\|_F dt \\
&\le 2e^K M_B M_X\sqrt{\ell^L(0)}\int_0^{t^*} e^{-\frac{\mu t}{2}} dt \\
&\le \frac{4e^K M_B M_X}{\mu}\sqrt{\ell^L(0)} \\
&\le \frac{M}{2}.
\end{aligned}$$

Finally, by (3.16),

$$\begin{aligned}\|B^L(t^*) - B(0)\|_F &\leq \int_0^{t^*} \Big\|\frac{dB^L}{dt}(t)\Big\|_F dt \\ &\leq \int_0^{t^*} \frac{2}{n}\sum_{i=1}^n \|(F^L(x_i;t) - y_i)h_{L,i}^L(t)^\top\|_F dt \\ &\leq 2e^K M_A M_X \sqrt{\ell^L(0)} \int_0^{t^*} e^{-\frac{\mu t}{2}} dt \\ &\leq \frac{4e^K M_A M_X}{\mu}\sqrt{\ell^L(0)} \\ &\leq \frac{M}{2},\end{aligned}$$

where the third inequality is a consequence of the Cauchy-Schwartz inequality and of the fact that $\|h_{L,i}^L(t)\| \leq e^K M_A M_X$. By continuity of A^L, Z_k^L, and B^L, these three bounds contradict the definition of t^*. We conclude that $T_{\max} = \infty$ and that the parameters stay within a ball of radius M of their initialization, yielding the inequalities, for $t \in \mathbb{R}_+$, $L \in \mathbb{N}^*$, and $k \in \{1, \dots, L\}$,

$$\|A^L(t)\|_F \leq M_A, \quad \|B^L(t)\|_F \leq M_B, \quad \|Z_k^L(t)\| \leq M_Z.$$

This proves statement (i) of the proposition. Moreover, the analysis above show that the derivatives of A^L, Z_k^L, and B^L are bounded by a bounded integrable function independent of L and k. This shows (iii), together with the fact that the functions $A^L(t)$, $Z_k^L(t)$, and $B^L(t)$ admit limits as $t \to \infty$. Furthermore, the convergence towards their limit is uniform over L and k, as we show for example for $A^L(t)$. If we denote by A_∞^L its limit, and apply the same steps as for bounding $\|A^L(t^*) - A^L(0)\|_F$, we obtain, for any $t \geq 0$,

$$\begin{aligned}\|A_\infty^L - A^L(t)\|_F &\leq \int_t^\infty \Big\|\frac{dA^L}{d\tau}(\tau)\Big\|_F d\tau \\ &\leq 2e^K M_B M_X \sqrt{\ell^L(0)} \int_t^\infty e^{\frac{-\mu\tau}{2}} d\tau \\ &= \frac{4e^K M_B M_X}{\mu} e^{\frac{-\mu t}{2}} \sqrt{\ell^L(0)} \\ &\leq \frac{M}{2} e^{\frac{-\mu t}{2}},\end{aligned}$$

where the last inequality comes from the definition of μ. The bound is independent of L, proving statement (iv). Statement (v) readily follows from (3.24).

To complete the proof, it remains to prove statement (ii) by bounding the differences $\|Z_{k+1}^L(t) - Z_k^L(t)\|$. Now that we know that the weights are bounded, we can follow the same steps as in the proof of Proposition 3.8 and show the existence of C_1, $C_2 > 0$ such that

$$\Big\|\frac{dZ_{k+1}^L}{dt}(t) - \frac{dZ_k^L}{dt}(t)\Big\| \leq \frac{1}{n}\sum_{i=1}^n \|p_{k,i}^L(t)\|\Big(\frac{C_1}{L} + C_2\|Z_{k+1}^L(t) - Z_k^L(t)\|\Big).$$

Using (3.25), we obtain

$$\Big\|\frac{dZ_{k+1}^L}{dt}(t) - \frac{dZ_k^L}{dt}(t)\Big\| \leq 2e^K M_B e^{-\frac{\mu t}{2}}\sqrt{\ell^L(0)}\Big(\frac{C_1}{L} + C_2\|Z_{k+1}^L(t) - Z_k^L(t)\|\Big).$$

Integrating between 0 and t, we obtain

$$\begin{aligned}\|Z_{k+1}^L(t) - Z_k^L(t)\| &\leq \|Z_{k+1}^L(0) - Z_k^L(0)\| + \int_0^t 2e^K M_B e^{-\frac{\mu\tau}{2}} \sqrt{\ell^L(0)} \frac{C_1}{L} d\tau \\ &\quad + \int_0^t 2e^K M_B e^{-\frac{\mu\tau}{2}} \sqrt{\ell^L(0)} C_2 \|Z_{k+1}^L(\tau) - Z_k^L(\tau)\| d\tau \\ &\leq \|Z_{k+1}^L(0) - Z_k^L(0)\| + \frac{C_1 M}{2M_X L} \\ &\quad + \int_0^t 2e^K M_B e^{-\frac{\mu\tau}{2}} \sqrt{\ell^L(0)} C_2 \|Z_{k+1}^L(\tau) - Z_k^L(\tau)\| d\tau,\end{aligned}$$

where the second inequality uses the definition of μ. By Grönwall's inequality,

$$\begin{aligned}\|Z_{k+1}^L(t) - Z_k^L(t)\| &\leq \Big(\|Z_{k+1}^L(0) - Z_k^L(0)\| + \frac{C_1 M}{2M_X L}\Big) \exp\Big(\int_0^t 2e^K M_B e^{-\frac{\mu\tau}{2}} \sqrt{\ell^L(0)} C_2 d\tau\Big) \\ &\leq \Big(\|Z_{k+1}^L(0) - Z_k^L(0)\| + \frac{C_1 M}{2M_X L}\Big) \exp\Big(\frac{C_2 M}{2M_X}\Big),\end{aligned}$$

again by definition of μ. Finally, since $Z_k^L(0) = Z^{\text{init}}(\frac{k}{L})$ and Z^{init} is Lipschitz continuous, this proves the existence of $\tilde{K} > 0$ (independent of L, t and k) such that $\|Z_{k+1}^L(t) - Z_k^L(t)\| \leq \frac{\tilde{K}}{L}$, which yields statement (ii). □

3.A.3 Generalized Arzelà–Ascoli theorem

Proposition 3.10 (Generalized Arzelà–Ascoli theorem)**.** *Let $I \subseteq \mathbb{R}_+$ be an interval. We denote by $(Z_k^L)_{L\in\mathbb{N}^*, 1\leq k\leq L}$ be a family of $\mathcal{C}^1$ functions from I to $\mathbb{R}^p$. Define*

$$\mathcal{Z}^L : [0,1] \times I \to \mathbb{R}^p,\ (s,t) \mapsto \mathcal{Z}^L(s,t) = Z^L_{\lfloor (L-1)s \rfloor + 1}(t).$$

Assume that there exist a constant $C > 0$ and a bounded integrable function $b : I \to \mathbb{R}$ such that the following statements hold for any $t \in I$ and $L \in \mathbb{N}^*$:

(i) For $k \in \{1, \dots, L-1\}$, $\|Z_{k+1}^L(t) - Z_k^L(t)\| \leq \frac{C}{L}$,

(ii) For $k \in \{1, \dots, L\}$, $\|Z_k^L(t)\| \leq C$ and $\|\frac{dZ_k^L}{dt}(t)\| \leq b(t)$.

Then there exist a subsequence $(\mathcal{Z}^{\varphi(L)})_{L\in\mathbb{N}^*}$ of $(\mathcal{Z}^L)_{L\in\mathbb{N}^*}$ and a Lipschitz continuous function $\mathcal{Z}^\varphi : [0,1] \times I \to \mathbb{R}^p$ such that $\mathcal{Z}^{\varphi(L)}(s,t)$ tends to $\mathcal{Z}^\varphi(s,t)$ uniformly over s and t.

Note that if I is a compact interval, then the existence of a (uniformly) convergent subsequence is guaranteed by the standard Arzelà–Ascoli theorem. Indeed, the uniform equicontinuity is a consequence of assumptions (i) and (ii), while (ii) provides a uniform bound. However, if I is not compact, more involved arguments are needed.

Proof. Assume, without loss of generality, that b is also bounded by C. According to assumption (i), for $t \in I$ and $i, j \in \{1, \dots, L\}$,

$$\|Z_i^L(t) - Z_j^L(t)\| \leq \frac{C|i-j|}{L}.$$

Also, according to (ii), for $t, t' \in I$ and $k \in \{1, \dots, L\}$,

$$\|Z_k^L(t) - Z_k^L(t')\| = \Big\| \int_{t'}^t \frac{dZ_k^L}{d\tau}(\tau) d\tau \Big\| \leq C|t - t'|.$$

It follows that, for $s, s' \in [0,1]$ and $t, t' \in I$,

$$\begin{aligned}\|\mathcal{Z}^L(s,t) - \mathcal{Z}^L(s',t')\| &\leq \|\mathcal{Z}^L(s,t) - \mathcal{Z}^L(s,t')\| + \|\mathcal{Z}^L(s,t') - \mathcal{Z}^L(s',t')\| \\ &\leq C|t-t'| + \frac{C|\lfloor (L-1)s \rfloor - \lfloor (L-1)s' \rfloor|}{L}.\end{aligned}$$

Therefore, with some simple algebra, we obtain

$$\|\mathcal{Z}^L(s,t) - \mathcal{Z}^L(s',t')\| \leq C|t-t'| + C|s-s'| + \frac{C}{L}. \tag{3.26}$$

The statement of the proposition is then a consequence of the next three steps.

There exists a convergent subsequence of $(\mathcal{Z}^L(s,t))_{L\in\mathbb{N}^*}$. First, let $((s_i,t_i))_{i\in\mathbb{N}} = (\mathbb{Q}\cap[0,1]) \times (\mathbb{Q}\cap I)$. By (ii), the sequence $(\mathcal{Z}^L(s_i,t_i))_{L\in\mathbb{N}^*,i\in\mathbb{N}}$ is bounded. It is therefore possible to construct by a diagonal procedure a subsequence $(\mathcal{Z}^{\varphi(L)})_{L\in\mathbb{N}^*}$ such that, for each $i \in \mathbb{N}$, $(\mathcal{Z}^{\varphi(L)}(s_i,t_i))_{L\in\mathbb{N}^*}$ is a convergent sequence.

Let us now show that $(\mathcal{Z}^{\varphi(L)}(s,t))_{L\in\mathbb{N}^*}$ converges for any $s \in [0,1]$ and $t \in I$, by proving that it is a Cauchy sequence in the complete metric space $\mathbb{R}^p$. Let $\varepsilon > 0$, $s \in [0,1]$, and $t \in I$. Since $((s_i,t_i))_{i\in\mathbb{N}}$ is dense in $[0,1]\times I$, there exists some $j \in \mathbb{N}$ such that $|s_j - s| \leq \varepsilon$ and $|t_j - t| \leq \varepsilon$. Then, for $L, M \in \mathbb{N}^*$, we have

$$\begin{aligned}&\|\mathcal{Z}^{\varphi(L)}(s,t) - \mathcal{Z}^{\varphi(M)}(s,t)\| \\ &\qquad\leq \|\mathcal{Z}^{\varphi(L)}(s,t) - \mathcal{Z}^{\varphi(L)}(s_j,t_j)\| + \|\mathcal{Z}^{\varphi(L)}(s_j,t_j) - \mathcal{Z}^{\varphi(M)}(s_j,t_j)\| \\ &\qquad\quad + \|\mathcal{Z}^{\varphi(M)}(s_j,t_j) - \mathcal{Z}^{\varphi(M)}(s,t)\| \\ &\qquad\leq 2C\varepsilon + \frac{C}{\varphi(L)} + \|\mathcal{Z}^{\varphi(L)}(s_j,t_j) - \mathcal{Z}^{\varphi(M)}(s_j,t_j)\| + 2C\varepsilon + \frac{C}{\varphi(M)},\end{aligned}$$

where we used inequality (3.26) twice. Since $(\mathcal{Z}^{\varphi(L)}(s_j,t_j))_{L\in\mathbb{N}^*}$ is a convergent sequence, it is a Cauchy sequence. Thus, the bound can be made arbitrarily small for L, M large enough. This shows that $(\mathcal{Z}^{\varphi(L)}(s,t))_{L\in\mathbb{N}^*}$ is also a Cauchy sequence. It is therefore convergent, and we denote by $\mathcal{Z}^{\varphi}(s,t)$ its limit.

The function $\mathcal{Z}^{\varphi}$ is Lipschitz continuous. By considering (3.26) for the subsequence $\varphi(L)$ and letting $L \to \infty$, we have that, for any $s, s' \in [0,1]$ and $t, t' \in I$,

$$\|\mathcal{Z}^{\varphi}(s,t) - \mathcal{Z}^{\varphi}(s',t')\| \leq C(|s-s'| + |t-t'|). \tag{3.27}$$

The convergence of $(\mathcal{Z}^{\varphi(L)}(s,t))_{L\in\mathbb{N}^*}$ to $\mathcal{Z}^{\varphi}(s,t)$ is uniform over s and t. Let $\varepsilon > 0$, $s \in [0,1]$, and $t \in I$. Then, by (3.26) and (3.27), it is possible to find $\delta > 0$ such that, for any $s', s'' \in [0,1]$ and $t', t'' \in I$ satisfying $|s' - s''| \leq \delta$ and $|t' - t''| \leq \delta$,

$$\|\mathcal{Z}^{\varphi(L)}(s',t') - \mathcal{Z}^{\varphi(L)}(s'',t')\| \leq \varepsilon + \frac{C}{\varphi(L)} \quad \text{and} \quad \|\mathcal{Z}^{\varphi}(s',t') - \mathcal{Z}^{\varphi}(s'',t')\| \leq \varepsilon, \tag{3.28}$$

and

$$\|\mathcal{Z}^{\varphi(L)}(s',t') - \mathcal{Z}^{\varphi(L)}(s',t'')\| \leq \varepsilon + \frac{C}{\varphi(L)} \quad \text{and} \quad \|\mathcal{Z}^{\varphi}(s',t') - \mathcal{Z}^{\varphi}(s',t'')\| \leq \varepsilon. \tag{3.29}$$

Furthermore, there exists a finite set $\{s_1, \dots, s_S\} \subset [0,1]$ such that

$$[0,1] \subset \bigcup_{i=1}^{S} (s_i - \delta, s_i + \delta).$$

In the sequel, we denote by s^* an element of $\{s_1, \dots, s_S\}$ that is at distance at most δ from s.

If I is unbounded, then, by assumption (ii) and since b is integrable, there exists some $t_0 > 0$ such that, for $t \geq t_0$,

$$\|\mathcal{Z}^{\varphi(L)}(s,t) - \mathcal{Z}^{\varphi(L)}(s,t_0)\| \leq \int_{t_0}^{t} \left\| \frac{d}{dt} Z^{\varphi(L)}_{\lfloor (\varphi(L)s-1) \rfloor + 1}(\tau) \right\| d\tau \leq \int_{t_0}^{t} b(\tau) d\tau \leq \varepsilon. \tag{3.30}$$

The same inequality holds for $\mathcal{Z}^{\varphi}$ by letting L tend to infinity. If I is bounded, we simply let $t_0 = \sup I$.

We may then pick a finite set $\{t_1, \dots, t_T\} \subset [0, t_0]$ such that

$$[0,t_0] \subset \bigcup_{i=1}^{T} (t_i - \delta, t_i + \delta).$$

Two cases may arise depending on the value of t. If $t \in [0,t_0]$, then there exists an element of the set $\{t_1, \dots, t_T\}$ at distance at most δ from t, and we denote it by t^*. If $t > t_0$, we let $t^* = t_0$. According to (3.29) and (3.30), we then have in both cases that

$$\|\mathcal{Z}^{\varphi(L)}(s,t) - \mathcal{Z}^{\varphi(L)}(s,t^*)\| \leqslant \varepsilon + \frac{C}{\varphi(L)} \quad \text{and} \quad \|\mathcal{Z}^{\varphi}(s,t) - \mathcal{Z}^{\varphi}(s,t^*)\| \leqslant \varepsilon. \tag{3.31}$$

To conclude, we have to bound the term $\|\mathcal{Z}^{\varphi(L)}(s,t) - \mathcal{Z}^{\varphi}(s,t)\|$ uniformly over s and t. We first have

$$\begin{aligned}
&\|\mathcal{Z}^{\varphi(L)}(s,t) - \mathcal{Z}^{\varphi}(s,t)\| \\
&\qquad \leq \|\mathcal{Z}^{\varphi(L)}(s,t) - \mathcal{Z}^{\varphi(L)}(s,t^*)\| + \|\mathcal{Z}^{\varphi(L)}(s,t^*) - \mathcal{Z}^{\varphi}(s,t^*)\| \\
&\qquad\quad + \|\mathcal{Z}^{\varphi}(s,t^*) - \mathcal{Z}^{\varphi}(s,t)\| \\
&\qquad \leq 2\varepsilon + \frac{C}{\varphi(L)} + \|\mathcal{Z}^{\varphi(L)}(s,t^*) - \mathcal{Z}^{\varphi}(s,t^*)\|,
\end{aligned}$$

where the last inequality is a consequence of (3.31). The last term can be bounded as follows:

$$\begin{aligned}
&\|\mathcal{Z}^{\varphi(L)}(s,t^*) - \mathcal{Z}^{\varphi}(s,t^*)\| \\
&\qquad \leq \|\mathcal{Z}^{\varphi(L)}(s,t^*) - \mathcal{Z}^{\varphi(L)}(s^*,t^*)\| + \|\mathcal{Z}^{\varphi(L)}(s^*,t^*) - \mathcal{Z}^{\varphi}(s^*,t^*)\| \\
&\qquad\quad + \|\mathcal{Z}^{\varphi}(s^*,t^*) - \mathcal{Z}^{\varphi}(s,t^*)\| \\
&\qquad \leq 2\varepsilon + \frac{C}{\varphi(L)} + \max_{i \in \{1,\dots,S\}} \|\mathcal{Z}^{\varphi(L)}(s_i,t^*) - \mathcal{Z}^{\varphi}(s_i,t^*)\|,
\end{aligned}$$

by using (3.28) and the fact that $s^* \in \{s_1, \dots, s_S\}$. Putting all the pieces together, we finally obtain

$$\|\mathcal{Z}^{\varphi(L)}(s,t) - \mathcal{Z}^{\varphi}(s,t)\| \leq 4\varepsilon + \frac{2C}{\varphi(L)} + \max_{i \in \{1,\dots,S\}, j \in \{1,\dots,T\}} \|\mathcal{Z}^{\varphi(L)}(s_i,t_j) - \mathcal{Z}^{\varphi}(s_i,t_j)\|.$$

By taking L large enough, independent of s and t, the sum of the last two terms can be made less than ε. Since ε is arbitrary, this concludes the proof. □

A consequence of this result is a simplified version for sequences of functions only indexed by L and not k, as follows.

Corollary 3.11. *Let $I \subseteq \mathbb{R}_+$ be an interval, and $(Z^L)_{L\in\mathbb{N}^*}$ be a family of $\mathcal{C}^1$ functions from I to $\mathbb{R}^p$. Assume that there exist a constant $C > 0$ and a bounded integrable function $b : I \to \mathbb{R}$ such that, for any $t \in I$ and $L \in \mathbb{N}^*$, $\|Z^L(t)\| \leq C$ and $\|\frac{dZ^L}{dt}(t)\| \leq b(t)$. Then there exist a subsequence $(Z^{\varphi(L)})_{L\in\mathbb{N}^*}$ of $(Z^L)_{L\in\mathbb{N}^*}$ and a function $Z^\varphi : I \to \mathbb{R}^p$ such that $Z^{\varphi(L)}(t)$ tends to $Z^\varphi(t)$ uniformly over t.*

3.A.4 Consistency of the Euler scheme for parameterized ODEs

Proposition 3.12 (Consistency of the Euler scheme for parameterized ODEs.)**.** *We denote by $(\theta_k^L)_{L\in\mathbb{N}^*,1\leq k\leq L}$ be a bounded family of vectors of $\mathbb{R}^p$, and let*

$$\Theta^L : [0,1] \to \mathbb{R}^p,\ s \mapsto \theta^L_{\lfloor (L-1)s\rfloor+1}.$$

Assume that there exists $\Theta : [0,1] \to \mathbb{R}^p$ a Lipschitz continuous function such that $\Theta^L(s)$ tends to $\Theta(s)$ uniformly over s. Let $(a^L)_{L\in\mathbb{N}^*}$ be a sequence of vectors in some compact $E \subset \mathbb{R}^d$ converging to $a \in E$. Let $g : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^d$ be a $\mathcal{C}^1$ function such that $g(0,\cdot) \equiv 0$ and $g(\cdot,\theta)$ is uniformly Lipschitz continuous for θ in any compact of $\mathbb{R}^p$. Consider the discrete scheme

$$\begin{aligned} u_0^L &= a^L \\ u_{k+1}^L &= u_k^L + \frac{1}{L} g(u_k^L, \theta_{k+1}^L), \quad k \in \{0,\dots,L-1\}. \end{aligned} \tag{3.32}$$

Then $u^L_{\lfloor Ls\rfloor}$ tends to $U(s)$ uniformly over $s \in [0,1]$, where U is the unique solution of the ODE

$$\begin{aligned} U(0) &= a \\ \frac{dU}{ds}(s) &= g(U(s), \Theta(s)), \quad s \in [0,1]. \end{aligned} \tag{3.33}$$

Moreover, the convergence only depends on the sequence $(a^L)_{L\in\mathbb{N}^*}$ and on its limit $a \in E$ through $(\|a^L - a\|)_{L\in\mathbb{N}^*}$.

Proof. Let M be a bound of the sequence $(\theta_k^L)_{L\in\mathbb{N}^*,1\leq k\leq L}$. By definition of Θ^L, the sequence $(\Theta^L)_{L\in\mathbb{N}^*}$ is also uniformly bounded by M, and the same is true for Θ. Then the function $g(\cdot,\Theta(s))$ is uniformly Lipschitz for $s \in [0,1]$. Furthermore, $(U,s) \mapsto g(U,\Theta(s))$ is continuous in s because g and Θ are continuous. Thus the ODE (3.33) has a unique solution on $[0,1]$ by the Picard-Lindelöf theorem (see Lemma 3.19).

Denote by C the uniform Lipschitz constant of $g(\cdot,\theta)$ for $\|\theta\| \leq M$. Since $g(0,\cdot) \equiv 0$ and $g(\cdot,\Theta(s))$ is C-Lipschitz, one has

$$\left\|\frac{dU}{ds}(s)\right\| = \|g(U(s),\Theta(s))\| \leq C\|U(s)\|.$$

Therefore, by Grönwall's inequality,

$$\|U(s)\| \leq \|U(0)\|\exp(C) = \|a\|\exp(C) \leq D_E \exp(C),$$

where $D_E = \sup_{x\in E}\|x\| < \infty$. A similar reasoning applies to the discrete scheme (3.32), using the discrete version of Grönwall's inequality. More precisely, for any $k \in \{0,\dots,L-1\}$,

$$\|u_{k+1}^L\| \leq \|u_k^L\| + \frac{1}{L}\|g(u_k^L,\theta_{k+1}^L)\| \leq \left(1 + \frac{C}{L}\right)\|u_k^L\|.$$

Thus,

$$\|u_k^L\| \leq \|u_0^L\| \exp(C) = \|a^L\| \exp(C) \leq D_E \exp(C).$$

Overall, we can consider a restriction of g to a compact set depending only on M, C, and E, which we will still denote by g with a slight abuse of notation.Since g is $\mathcal{C}^1$, it is therefore bounded and Lipschitz continuous, and we still let C be its Lipschitz constant.

For $L \in \mathbb{N}^*$ and $k \in \{0, \dots, L\}$, we denote by Δ_k^L the gap between the continuous and the discrete schemes, i.e.,

$$\Delta_k^L = \Big\| U\Big(\frac{k}{L}\Big) - u_k^L \Big\|.$$

The next step is to recursively bound the size of this gap, first observing that $\Delta_0^L = \|a^L - a\|$. We have that

$$s \mapsto \frac{dU}{ds}(s) = g(U(s), \Theta(s)) \tag{3.34}$$

is a Lipschitz continuous function with some Lipschitz constant $\tilde{C}$. To see this, just note that U itself is Lipschitz continuous in s, since g is bounded, and therefore the function (3.34) is a composition of Lipschitz continuous functions. In particular, $\frac{dU}{ds}$ is almost everywhere differentiable, and its derivative $\frac{d^2U}{ds^2}(s)$ is bounded in the supremum norm by $\tilde{C}$. As a consequence, for $k \in \{0, \dots, L-1\}$, the Taylor expansion of U on $[\frac{k}{L}, \frac{k+1}{L}]$ takes the form

$$U\Big(\frac{k+1}{L}\Big) = U\Big(\frac{k}{L}\Big) + \frac{1}{L}\frac{dU}{ds}\Big(\frac{k}{L}\Big) + \int_{k/L}^{(k+1)/L} \Big(\frac{k+1}{L} - s\Big)\frac{d^2U}{ds^2}(s)ds,$$

where the norm of the remainder term is less than $\tilde{C}/L^2$. Therefore,

$$\begin{aligned}
\Delta_{k+1}^L &= \Big\| U\Big(\frac{k+1}{L}\Big) - u_{k+1}^L \Big\| \\
&= \Big\| U\Big(\frac{k}{L}\Big) + \frac{1}{L} g\Big(U\Big(\frac{k}{L}\Big), \Theta\Big(\frac{k}{L}\Big)\Big) + \int_{k/L}^{(k+1)/L} \Big(\frac{k+1}{L} - s\Big)\frac{d^2U}{ds^2}(s)ds \\
&\quad - u_k^L - \frac{1}{L} g(u_k^L, \theta_{k+1}^L) \Big\| \\
&\leq \Big\| U\Big(\frac{k}{L}\Big) - u_k^L \Big\| + \Big\| \frac{1}{L} g\Big(U\Big(\frac{k}{L}\Big), \Theta\Big(\frac{k}{L}\Big)\Big) - \frac{1}{L} g(u_k^L, \theta_{k+1}^L) \Big\| \\
&\quad + \int_{k/L}^{(k+1)/L} \Big(\frac{k+1}{L} - s\Big) \Big\| \frac{d^2U}{ds^2}(s) \Big\| ds \\
&\leqslant \Delta_k^L + \frac{C}{L}\Delta_k^L + \frac{C}{L} \Big\| \Theta\Big(\frac{k}{L}\Big) - \theta_{k+1}^L \Big\| + \frac{\tilde{C}}{L^2}.
\end{aligned}$$

In the last inequality, we used the fact that g is C-Lipschitz. Since,by definition, $\theta_{k+1}^L = \Theta^L(\frac{k}{L-1})$, we obtain, for $k \in \{0, \dots, L-1\}$,

$$\begin{aligned}
\Delta_{k+1}^L &\leqslant \Big(1 + \frac{C}{L}\Big)\Delta_k^L + \frac{C}{L} \Big\| \Theta\Big(\frac{k}{L}\Big) - \Theta^L\Big(\frac{k}{L-1}\Big) \Big\| + \frac{\tilde{C}}{L^2} \\
&\leqslant \Big(1 + \frac{C}{L}\Big)\Delta_k^L + \frac{C}{L} \sup_{s \in [0,1]} \|\Theta(s) - \Theta^L(s)\| + \frac{C}{L} \Big\| \Theta\Big(\frac{k}{L}\Big) - \Theta\Big(\frac{k}{L-1}\Big) \Big\| + \frac{\tilde{C}}{L^2} \\
&\leqslant \Big(1 + \frac{C}{L}\Big)\Delta_k^L + \frac{C}{L} \sup_{s \in [0,1]} \|\Theta(s) - \Theta^L(s)\| + \frac{CC_\Theta}{L^2} + \frac{\tilde{C}}{L^2},
\end{aligned}$$

where C_Θ is the Lipschitz constant of Θ. By the discrete Grönwall's inequality, we deduce that, for $k \in \{0, \dots, L-1\}$,

$$\begin{aligned}\Delta_{k+1}^L &\leqslant \Big(\Delta_0^L + \sup_{s\in[0,1]} \|\Theta(s) - \Theta^L(s)\| + \frac{C_\Theta}{L} + \frac{\tilde{C}}{LC}\Big)e^C \\ &= \Big(\|a^L - a\| + \sup_{s\in[0,1]} \|\Theta(s) - \Theta^L(s)\| + \frac{C_\Theta}{L} + \frac{\tilde{C}}{LC}\Big)e^C. \end{aligned} \tag{3.35}$$

This shows that the gaps Δ_k^L converge to zero uniformly over $k \in \{0, \dots, L\}$ as L tends to infinity.

We conclude by observing that, for any $s \in [0,1]$,

$$\|U(s) - u^L_{\lfloor Ls \rfloor}\| \leqslant \Big\|U(s) - U\Big(\frac{\lfloor Ls \rfloor}{L}\Big)\Big\| + \Big\|U\Big(\frac{\lfloor Ls \rfloor}{L}\Big) - u^L_{\lfloor Ls \rfloor}\Big\| \leq \frac{C_U}{L} + \Delta^L_{\lfloor Ls \rfloor}, \tag{3.36}$$

where C_U is the Lipschitz constant of U. Both terms converge to zero uniformly over s as L tends to infinity. Finally, an inspection of our bounds shows that the convergence only depends on $(a^L)_{L\in\mathbb{N}^*} \in E^{\mathbb{N}^*}$ through $\|a^L - a\|$. □

The results of Proposition 3.12 can be extended without much effort to two other related cases. First, the parameters θ_k^L may depend on some other variable t, as long as all assumptions are verified uniformly over t. Second, these parameters may converge to some limit parameters as both L and t go to infinity. This is encapsulated in the following two corollaries.

Corollary 3.13. *Let $I \subseteq \mathbb{R}_+$ be an interval. Let $(\theta_k^L)_{L\in\mathbb{N}^*, 1\leq k\leq L}$ be a uniformly bounded family of functions from I to $\mathbb{R}^p$, and let*

$$\Theta^L : [0,1] \times I \to \mathbb{R}^p,\ (s,t) \mapsto \theta^L_{\lfloor (L-1)s \rfloor + 1}(t).$$

Assume that there exists a function $\Theta : [0,1] \times I \to \mathbb{R}^p$ such that $\Theta^L(s,t)$ tends to $\Theta(s,t)$ uniformly over s and t, and $\Theta(\cdot, t)$ is uniformly Lipschitz continuous for $t \in I$. Let $(a^L)_{L\in\mathbb{N}^*}$ be a family of functions from I to some compact $E \subset \mathbb{R}^d$, uniformly converging to $a : I \to E$. Let $g : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^d$ be a $\mathcal{C}^1$ function such that $g(0,\cdot) \equiv 0$ and $g(\cdot, \theta)$ is uniformly Lipschitz continuous for θ in any compact of $\mathbb{R}^p$. Consider the discrete scheme, for $t \in I$,

$$\begin{aligned} u_0^L(t) &= a^L(t) \\ u_{k+1}^L(t) &= u_k^L(t) + \frac{1}{L} g(u_k^L(t), \theta_{k+1}^L(t)), \quad k \in \{0, \dots, L-1\}. \end{aligned}$$

Then $u^L_{\lfloor Ls \rfloor}(t)$ tends to $U(s,t)$ uniformly over $s \in [0,1]$ and $t \in I$, where $U(\cdot, t)$ is the unique solution of the ODE

$$\begin{aligned} U(0,t) &= a(t) \\ \frac{\partial U}{\partial s}(s,t) &= g(U(s,t), \Theta(s,t)), \quad s \in [0,1]. \end{aligned}$$

Moreover, the convergence only depends on the sequence $(a^L)_{L\in\mathbb{N}^*}$ and on its limit $a \in E^I$ through $(\sup_{t\in I} \|a^L(t) - a(t)\|)_{L\in\mathbb{N}^*}$.

Corollary 3.14. *Let $I \subseteq \mathbb{R}_+$ be an interval. Let $(\theta_k^L)_{L\in\mathbb{N}^*, 1\leq k\leq L}$ be a uniformly bounded family of functions from I to $\mathbb{R}^p$, and let*

$$\Theta^L : [0,1] \times \mathbb{R}_+ \to \mathbb{R}^p,\ (s,t) \mapsto \theta^L_{\lfloor (L-1)s \rfloor + 1}(t).$$

Assume that there exists a function $\Theta_\infty : [0,1] \to \mathbb{R}^p$ such that $\Theta^L(s,t)$ tends to $\Theta_\infty(s)$ uniformly over s as $L, t \to \infty$, and Θ_∞ is Lipschitz continuous. Let $(a^L)_{L\in\mathbb{N}^*}$ be a family of functions from I to some compact $E \subset \mathbb{R}^d$, and converging to $a_\infty \in E$ as $L, t \to \infty$. Let $g : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^d$ be a $\mathcal{C}^1$ function such that $g(0,\cdot) \equiv 0$ and $g(\cdot,\theta)$ is uniformly Lipschitz continuous for θ in any compact of $\mathbb{R}^p$. Consider the discrete scheme, for $t \in I$,

$$\begin{aligned} u_0^L(t) &= a^L(t) \\ u_{k+1}^L(t) &= u_k^L(t) + \frac{1}{L} g(u_k^L(t), \theta_{k+1}^L(t)), \quad k \in \{0, \dots, L-1\}. \end{aligned}$$

Then $u_{\lfloor Ls \rfloor}^L(t)$ tends to $U(s)$ uniformly over $s \in [0,1]$ as $L, t \to \infty$, where U is the unique solution of the ODE

$$\begin{aligned} U(0) &= a_\infty \\ \frac{dU}{ds}(s) &= g(U(s), \Theta_\infty(s)), \quad s \in [0,1]. \end{aligned}$$

Moreover, the convergence only depends on the sequence $(a^L)_{L\in\mathbb{N}^*}$ and on its limit $a \in E^I$ through $(\sup_{t\in I} \|a^L(t) - a(t)\|)_{L\in\mathbb{N}^*}$.

3.A.5 Large-depth convergence of the gradient flow

This section is devoted to proving the main result of Appendix 3.A, namely the large-depth convergence of the gradient flow. The setting we consider encompasses both Section 3.4.1 (finite training time and clipped gradient flow) and Section 3.4.2 (arbitrary training time and standard gradient flow). To this end, we consider a training interval $I = [0,T] \subseteq \mathbb{R}_+$, for $T \leq \infty$, and the gradient flow formulation (3.5), which is equivalent to the standard gradient flow (3.4) if π equals the identity. Note that we do not need to assume in the following proof that π is bounded (but only Lipschitz continuous). Therefore, the proof also holds in the case where π equals the identity.

Theorem 3.15. *Consider the residual network (3.12) initialized as explained in Appendix 3.A and trained with the gradient flow (3.5) on $I = [0,T] \subseteq \mathbb{R}_+$, for some $T \in (0,\infty]$. Assume that there exists a unique solution to the gradient flow, such that $(A^L)_{L\in\mathbb{N}^*}$ and $(B^L)_{L\in\mathbb{N}^*}$ each satisfies the assumptions of Corollary 3.11, and $(Z_k^L)_{L\in\mathbb{N}^*, 1\leq k\leq L}$ satisfies the assumptions of Proposition 3.10. Then the following four statements hold **as L tends to infinity**:*

(i) There exist functions $A : I \to \mathbb{R}^{q\times d}$ and $B : I \to \mathbb{R}^{d'\times q}$ such that $A^L(t)$ and $B^L(t)$ converge uniformly over $t \in I$ to $A(t)$ and $B(t)$.

(ii) There exists a Lipschitz continuous function $\mathcal{Z} : [0,1] \times I \to \mathbb{R}^p$ such that

$$\mathcal{Z}^L : [0,1] \times I \to \mathbb{R}^p, \ (s,t) \mapsto \mathcal{Z}^L(s,t) = Z_{\lfloor (L-1)s \rfloor + 1}^L(t)$$

converges uniformly over $s \in [0,1]$ and $t \in I$ to $\mathcal{Z}(s,t)$.

(iii) Uniformly over $s \in [0,1]$, $t \in I$, and $x \in \mathcal{X}$, the hidden layer $h_{\lfloor Ls \rfloor}^L(t)$ converges to the solution at time s of the neural ODE

$$\begin{aligned} H(0,t) &= A(t)x \\ \frac{\partial H}{\partial s}(s,t) &= f(H(s,t), \mathcal{Z}(s,t)), \quad s \in [0,1]. \end{aligned}$$

(iv) Uniformly over $t \in I$ and $x \in \mathcal{X}$, the output $F^L(x;t)$ converges to $B(t)H(1,t)$.

Proof. According to Proposition 3.10, there exists a subsequence $(\mathcal{Z}^{\varphi(L)})_{L\in\mathbb{N}^*}$ of $(\mathcal{Z}^L)_{L\in\mathbb{N}^*}$ and a Lipschitz continuous function $\mathcal{Z}^\varphi : [0,1]\times I \to \mathbb{R}^p$ such that $\mathcal{Z}^{\varphi(L)}(s,t)$ tends to $\mathcal{Z}^\varphi(s,t)$ uniformly over s and t. Similarly, by Corollary 3.11, there exists subsequences of $(A^L)_{L\in\mathbb{N}^*}$ and $(B^L)_{L\in\mathbb{N}^*}$ that converge uniformly. With a slight abuse of notation, we still denote these subsequences by φ, and the corresponding limits by A^φ and B^φ.

In the remainder, we prove the uniqueness of the accumulation point $(Z^\varphi, A^\varphi, B^\varphi)$ by showing that it is the solution of an ODE that satisfies the assumptions of the Picard-Lindelöf theorem. The statements (i) to (iv) then follow easily.

Consider a general input $(x,y)\in\mathcal{X}\times\mathcal{Y}$, and let $H^L(s,t) = h^L_{\lfloor Ls\rfloor}(t)$ (recall that $h^L_k(t)$ is defined by the forward propagation (3.12)). Corollary 3.13, with $\theta^L_k = Z^{\varphi(L)}_k$, $\Theta = \mathcal{Z}^\varphi$, $a^L = A^{\varphi(L)}x$, $g = f$, ensures that $H^{\varphi(L)}(s,t)$ converges uniformly (over s and t) to $H^\varphi(s,t)$ that is the solution at time s of the ODE

$$\begin{aligned} H^\varphi(0,t) &= A^\varphi(t)x \\ \frac{\partial H^\varphi}{\partial s}(s,t) &= f(H^\varphi(s,t), \mathcal{Z}^\varphi(s,t)), \quad s\in[0,1]. \end{aligned}$$

By inspecting the proof of the corollary, we also have that $(h^{\varphi(L)}_k)_{L\in\mathbb{N}^*,1\le k\le\varphi(L)}$ and $(H^{\varphi(L)})_{L\in\mathbb{N}^*}$ are uniformly bounded and that $H^\varphi(\cdot,t)$ is uniformly Lipschitz continuous for $t\in I$.

We now turn our attention to the backpropagation recurrence (3.13), which defines the backward state $p^L_k(t)$. First observe that the convergence of $H^{\varphi(L)}$ implies that

$$p^{\varphi(L)}_{\varphi(L)}(t) = 2B^{\varphi(L)}(t)^\top(B^{\varphi(L)}(t)h^{\varphi(L)}_{\varphi(L)}(t) - y) = 2B^{\varphi(L)}(t)^\top(B^{\varphi(L)}(t)H^{\varphi(L)}(1,t) - y)$$

converges uniformly to $2B^\varphi(t)^\top(B^\varphi(t)H^\varphi(1,t)-y)\in\mathbb{R}^d$. Now, let $P^L(s,t) = p^L_{\lfloor Ls\rfloor}(t)$. We apply again Corollary 3.13, this time to the backpropagation recurrence (3.13), with $\theta^L_k = (h^{\varphi(L)}_k, Z^{\varphi(L)}_k)$, $\Theta = (H^\varphi, \mathcal{Z}^\varphi)$, $g : (p,(h,Z)) \mapsto \partial_1 f(h,Z)p$, and $a^L = 2(B^{\varphi(L)})^\top(B^{\varphi(L)}H^{\varphi(L)}(1,\cdot) - y)$. Let us quickly check that the conditions of the corollary are met:

- The sequence $(h^{\varphi(L)}_k)_{L\in\mathbb{N}^*,1\le k\le\varphi(L)}$ is bounded, as noted previously, and the same holds for $(Z^{\varphi(L)}_k)_{L\in\mathbb{N}^*,1\le k\le\varphi(L)}$ by the assumptions of Theorem 3.15.
- The function $H^\varphi(\cdot,t)$ is uniformly Lipschitz continuous for $t\in I$, as noted previously, and the same is true for $Z^\varphi(\cdot,t)$ since Z^φ is Lipschitz continuous.
- The function $h^{\varphi(L)}_{\lfloor(\varphi(L)-1)s\rfloor+1}(t)$ tends to $H^\varphi(s,t)$ uniformly over s and t, as seen in the beginning of the proof. More precisely, we know that $H^{\varphi(L)}(s,t) = h^{\varphi(L)}_{\lfloor\varphi(L)s\rfloor}(t)$ tends to $H^\varphi(s,t)$. Simple algebra and the fact that two successive iterates of (3.12) are separated by a distance proportional to $1/L$ show that both statements are equivalent. Furthermore, $\mathcal{Z}^{\varphi(L)}(s,t)$ tends to $\mathcal{Z}^\varphi(s,t)$ uniformly over s and t as noted above.
- The sequence $(a^L)_{L\in\mathbb{N}^*}$ is uniformly bounded, since $B^{\varphi(L)}$ and $H^{\varphi(L)}(1,\cdot)$ are. It also converges uniformly to $a : t\mapsto 2B^\varphi(t)^\top(B^\varphi(t)H^\varphi(1,t)-y)$.
- The function g is $\mathcal{C}^1$ since f is $\mathcal{C}^2$. We clearly have $g(0,\cdot)\equiv 0$. Finally, $g(\cdot,(h,Z))$ is uniformly Lipschitz continuous for (h,Z) in any compact since $\partial_1 f$ is continuous.

Overall, we obtain that $P^{\varphi(L)}(s,t)$ converges uniformly (over s and t) to $P^\varphi(s,t)$, the solution at

time s of the backward ODE

$$P^{\varphi}(1,t) = 2B^{\varphi}(t)^{\top}(B^{\varphi}(t)H^{\varphi}(1,t) - y)$$
$$\frac{\partial P^{\varphi}}{\partial s}(s,t) = \partial_1 f(H^{\varphi}(s,t), \mathcal{Z}^{\varphi}(s,t))P^{\varphi}(s,t), \quad s \in [0,1].$$

Furthermore, the proof of the corollary shows that $(P^{\varphi(L)})_{L\in\mathbb{N}^*}$ is uniformly bounded. Now, recall that the gradient flow for $Z_k^{\varphi(L)}(t)$, given by (3.5) and (3.15), takes the following form, for $t \in I$ and $k \in \{1, \dots, \varphi(L)\}$,

$$\frac{\partial Z_k^{\varphi(L)}(t)}{\partial t} = \pi\Big(-\frac{1}{n}\sum_{i=1}^{n} \partial_2 f(h_{k-1,i}^{\varphi(L)}(t), Z_k^{\varphi(L)}(t))^{\top} p_{k-1,i}^{\varphi(L)}(t)\Big),$$

where the i subscript corresponds to the i-th input x_i. By definition, for $s \in [0,1]$, $\mathcal{Z}^{\varphi(L)}(s,t) = Z^{\varphi(L)}_{\lfloor(\varphi(L)-1)s\rfloor+1}(t)$. Thus, the equation above can be rewritten, for $s \in [0,1]$ and $t \in I$,

$$\frac{\partial \mathcal{Z}^{\varphi(L)}(s,t)}{\partial t} = \pi\Big(-\frac{1}{n}\sum_{i=1}^{n} \partial_2 f(h^{\varphi(L)}_{\lfloor(\varphi(L)-1)s\rfloor,i}(t), Z^{\varphi(L)}_{\lfloor(\varphi(L)-1)s\rfloor+1}(t))^{\top} p^{\varphi(L)}_{\lfloor(\varphi(L)-1)s\rfloor,i}(t)\Big). \tag{3.37}$$

The term inside π can be rewritten as

$$-\frac{1}{n}\sum_{i=1}^{n} \partial_2 f\Big(H_i^{\varphi(L)}\Big(\frac{\lfloor(\varphi(L)-1)s\rfloor}{\varphi(L)}, t\Big), \mathcal{Z}^{\varphi(L)}(s,t)\Big)^{\top} P_i^{\varphi(L)}\Big(\frac{\lfloor(\varphi(L)-1)s\rfloor}{\varphi(L)}, t\Big).$$

Since f is $\mathcal{C}^2$, $\partial_2 f$ is locally Lipschitz continuous. Applying the first part of the proof to the specific case of x_i, we know that $H_i^{\varphi(L)}$ and $P_i^{\varphi(L)}$ uniformly bounded, and that $H_i^{\varphi(L)}(s,t)$ and $P_i^{\varphi(L)}(s,t)$ converge uniformly to $H_i^{\varphi}(s,t)$ and $P_i^{\varphi}(s,t)$. Therefore, the right-hand side of (3.37) converges uniformly over s and t to

$$\pi\Big(-\frac{1}{n}\sum_{i=1}^{n} \partial_2 f(H_i^{\varphi}(s,t), \mathcal{Z}^{\varphi}(s,t))^{\top} P_i^{\varphi}(s,t)\Big).$$

We have just shown the uniform convergence of the derivative in t of $\mathcal{Z}^{\varphi(L)}(s,t)$. Furthermore, we know that, for $s \in [0,1]$, the sequence $(t \mapsto \mathcal{Z}^{\varphi(L)}(s,t))_{L\in\mathbb{N}^*}$ converges to $\mathcal{Z}^{\varphi}(s,\cdot)$. These two statements imply that $\mathcal{Z}^{\varphi}$ is differentiable with respect to t and that, for $s \in [0,1]$, its derivative satisfies the ordinary differential equation

$$\frac{\partial \mathcal{Z}^{\varphi}(s,t)}{\partial t} = \pi\Big(-\frac{1}{n}\sum_{i=1}^{n} \partial_2 f(H_i^{\varphi}(s,t), \mathcal{Z}^{\varphi}(s,t))^{\top} P_i^{\varphi}(s,t)\Big). \tag{3.38}$$

Moreover, by our initialization scheme,

$$\mathcal{Z}^{\varphi}(s,0) = Z^{\text{init}}(s). \tag{3.39}$$

A similar approach reveals that $A^{\varphi}(t)$ and $B^{\varphi}(t)$ are differentiable and that they verify the equations

$$\frac{dA^{\varphi}}{dt}(t) = \pi\Big(-\frac{1}{n}\sum_{i=1}^{n} P_i^{\varphi}(0,t)x_i^{\top}\Big), \qquad A^{\varphi}(0) = A^{\text{init}}, \tag{3.40}$$

$$\frac{dB^{\varphi}}{dt}(t) = \pi\Big(-\frac{2}{n}\sum_{i=1}^{n} (B^{\varphi}(t)H_i^{\varphi}(1,t) - y_i)H_i^{\varphi}(1,t)^{\top}\Big), \qquad B^{\varphi}(0) = B^{\text{init}}. \tag{3.41}$$

The equations (3.38) to (3.41) can be seen as an initial value problem whose variables are the function $\mathcal{Z}^\varphi(\cdot,t):[0,1]\to\mathbb{R}^p$ and the matrices $A^\varphi(t)\in\mathbb{R}^{q\times d}, B^\varphi(t)\in\mathbb{R}^{d'\times q}$. To complete the proof, it remains to show, using the Picard-Lindelöf theorem (see Lemma 3.19), that there exists a unique solution to this problem. First, note that the space $\mathcal{B}([0,1],\mathbb{R}^p)$ of bounded functions from $[0,1]$ to $\mathbb{R}^p$ endowed with the supremum norm is a Banach space, which is the proper space in which to apply the Picard-Lindelöf theorem. We therefore endow the space of parameters $\mathcal{B}([0,1],\mathbb{R}^p)\times\mathbb{R}^{q\times d}\times\mathbb{R}^{d'\times q}$ with the norm

$$\|(\mathcal{Z},A,B)\| := \sup_{s\in[0,1]}\|\mathcal{Z}(s)\| + \|A\|_2 + \|B\|_2,$$

which makes it a Banach space. We have to show that the mapping

$$\begin{aligned}(\mathcal{Z},A,B)\mapsto\Big(s\mapsto\pi\Big(-\frac{1}{n}\sum_{i=1}^n\partial_2 f(H_i(s),\mathcal{Z}(s))^\top P_i(s)\Big),\\ \pi\Big(-\frac{1}{n}\sum_{i=1}^n P_i(0)x_i^\top\Big),\ \pi\Big(-\frac{2}{n}\sum_{i=1}^n(BH_i(1)-y_i)H_i(1)^\top\Big)\Big)\end{aligned} \tag{3.42}$$

is locally Lipschitz continuous with respect to this norm, where we recall that $H_i(s)$ in (3.42) is the solution at time s of the initial value problem

$$\begin{aligned}H_i(0) &= Ax_i\\ \frac{dH_i}{ds}(s) &= f(H_i(s),\mathcal{Z}(s)),\quad s\in[0,1],\end{aligned} \tag{3.43}$$

and $P_i(s)$ is the solution at time s of the initial value problem

$$\begin{aligned}P_i(1) &= 2B^\top(BH_i(1)-y_i)\\ \frac{dP_i}{ds}(s) &= \partial_1 f(H_i(s),\mathcal{Z}(s))P_i(s),\quad s\in[0,1].\end{aligned} \tag{3.44}$$

To prove that the mapping (3.42) is locally Lipschitz continuous, we first check that it is well defined. Since $\mathcal{Z}$ is assumed to be only bounded (and not continuous), the solutions of the initial value problems (3.43) and (3.44) are well defined in the sense of the Caratheodory conditions, which are given in Lemma 3.20.

Next, we can show that $(\mathcal{Z},A,B)\mapsto H_i$ is locally Lipschitz continuous for $i\in\{1,\dots,n\}$. To do this, consider two sets of parameters $(\mathcal{Z},A,B)$ and $(\tilde{\mathcal{Z}},\tilde{A},\tilde{B})$ belonging to a compact set D. Let H_i and $\tilde{H}_i$ denote the corresponding hidden states. As in the proof of Proposition 3.12, it holds that H_i and $\tilde{H}_i$ belong to some compact set E that depends only on D and f. Let K_f be the Lipschitz constant of the $\mathcal{C}^1$ function f on $E\times D$. Then,

$$\begin{aligned}\|\tilde{H}_i(s)-H_i(s)\| &\le \|\tilde{H}_i(0)-H_i(0)\| + \int_0^s\Big\|\frac{d\tilde{H}_i}{dr}(r)-\frac{dH_i}{dr}(r)\Big\|dr\\ &\le \|\tilde{H}_i(0)-H_i(0)\| + \int_0^s\|f(\tilde{H}_i(r),\tilde{\mathcal{Z}}(r))-f(H_i(r),\mathcal{Z}(r))\|dr.\end{aligned}$$

The norm inside the integral can be bounded by

$$\begin{aligned}\|f(\tilde{H}_i(r),\tilde{\mathcal{Z}}(r))-f(\tilde{H}_i(r),\mathcal{Z}(r))\| + \|f(\tilde{H}_i(r),\mathcal{Z}(r))-f(H_i(r),\mathcal{Z}(r))\|\\ \le K_f\sup_{r\in[0,1]}\|\tilde{\mathcal{Z}}(r)-\mathcal{Z}(r)\| + K_f\|\tilde{H}_i(r)-H_i(r)\|.\end{aligned}$$

Therefore,

$$\|\tilde{H}_i(s) - H_i(s)\| \leq \|\tilde{A} - A\|_2\|x_i\| + K_f \sup_{r\in[0,1]} \|\tilde{\mathcal{Z}}(r) - \mathcal{Z}(r)\| + \int_0^s K_f\|\tilde{H}_i(r) - H_i(r)\|dr.$$

Using Grönwall's inequality, we obtain, for any $s \in [0,1]$,

$$\|\tilde{H}_i(s) - H_i(s)\| \leq \Big(\|\tilde{A} - A\|_2\|x_i\| + K_f \sup_{r\in[0,1]} \|\tilde{\mathcal{Z}}(r) - \mathcal{Z}(r)\|\Big) \exp(K_f).$$

This shows that the function $(\mathcal{Z}, A, B) \mapsto H_i$ is locally Lipschitz continuous. One proves by similar arguments that the function $(\mathcal{Z}, A, B) \mapsto P_i$ is locally Lipschitz continuous. Thus, overall, the mapping (3.42) is locally Lipschitz continuous as a composition of locally Lipschitz continuous functions.

The Picard-Lindelöf theorem guarantees the uniqueness of the maximal solution of the initial value problem (3.38)–(3.41) in the space $\mathcal{B}([0,1],\mathbb{R}^p) \times \mathbb{R}^{d\times q} \times \mathbb{R}^{d'\times q}$. Since any accumulation point $(\mathcal{Z}^\varphi, A^\varphi, B^\varphi)$ is a solution belonging to this space, this proves the uniqueness of the accumulation point, which we therefore denote as $(\mathcal{Z}, A, B)$.

The uniform convergence of $(\mathcal{Z}^L, A^L, B^L)$ to $(\mathcal{Z}, A, B)$ is then easily shown by contradiction. Suppose that uniform convergence does not hold. If this is true, then there exists a subsequence that stays at distance $\varepsilon > 0$ from $(\mathcal{Z}, A, B)$ (in the sense of the uniform norm). Then arguments similar to the beginning of the proof show the existence of a second accumulation point, which is a contradiction. This shows the uniform convergence, yielding statements (i) and (ii) of the theorem.

Finally, reapplying Corollary 3.13 with $\theta_k^L = Z_k^L$, $\Theta = \mathcal{Z}$, $a^L = A^L x$, $g = f$, completes the proof by proving statements (iii) and (iv). □

Training dynamics of the limiting weights. Interestingly, the proof of Theorem 3.15 provides us with an explicit description of the evolution of the continuous-depth limiting weights during training. With the notation of the proof, the continuous weights satisfy the training dynamics:

$$\begin{aligned}
\frac{dA}{dt}(t) &= \pi\Big(-\frac{1}{n}\sum_{i=1}^n P_i(0,t)x_i^\top\Big)\\
\frac{\partial \mathcal{Z}}{\partial t}(s,t) &= \pi\Big(-\frac{1}{n}\sum_{i=1}^n \partial_2 f(H_i(s,t), \mathcal{Z}(s,t))^\top P_i(s,t)\Big)\\
\frac{dB}{dt}(t) &= \pi\Big(-\frac{2}{n}\sum_{i=1}^n (B(t)H_i(1,t) - y_i)H_i(1,t)^\top\Big),
\end{aligned}$$

where we recall that $H_i(s,t)$ is the solution at time s of the initial value problem

$$\begin{aligned}
H_i(0,t) &= A(t)x_i\\
\frac{\partial H_i}{\partial s}(s,t) &= f(H_i(s,t), \mathcal{Z}(s,t)), \quad s \in [0,1],
\end{aligned}$$

and $P_i(s,t)$ is the solution at time s of the problem

$$\begin{aligned}
P_i(1,t) &= 2B(t)^\top(B(t)H_i(1,t) - y_i)\\
\frac{\partial P_i}{\partial s}(s,t) &= \partial_1 f(H_i(s,t), \mathcal{Z}(s,t))P_i(s,t), \quad s \in [0,1].
\end{aligned}$$

These equations can be thought of as the continuous-depth equivalent of the backpropagation equations.

3.A.6 Existence of the double limit when L, t tend to infinity

Proposition 3.16. *Consider the residual network (3.12), and assume that:*

(i) $A^L(t)$, $Z^L_{\lfloor Ls \rfloor}(t)$, *and* $B^L(t)$ *converge uniformly over* $L \in \mathbb{N}^*$ *and* $s \in [0,1]$ *as* $t \to \infty$.

(ii) $A^L(t)$, $Z^L_{\lfloor Ls \rfloor}(t)$, *and* $B^L(t)$ *converge uniformly over* $t \in \mathbb{R}_+$ *and* $s \in [0,1]$ *as* $L \to \infty$.

(iii) The loss $\ell^L(t)$ *converges to* 0 *uniformly over* $L \in \mathbb{N}^*$ *as* $t \to \infty$.

Then the following four statements hold ***as*** t ***and*** L ***tend to infinity****:*

(i) There exist matrices $A_\infty \in \mathbb{R}^{q \times d}$ *and* $B_\infty \in \mathbb{R}^{d' \times q}$ *such that* $A^L(t)$ *and* $B^L(t)$ *converge to* A_∞ *and* B_∞.

(ii) There exists a Lipschitz continuous function $\mathcal{Z}_\infty : [0,1] \to \mathbb{R}^p$ *such that* $Z^L_{\lfloor Ls \rfloor}(t)$ *converges to* $\mathcal{Z}_\infty(t)$ *uniformly over* $s \in [0,1]$.

(iii) Uniformly over $s \in [0,1]$ *and* $x \in \mathcal{X}$, *the hidden layer* $h^L_{\lfloor Ls \rfloor}(t)$ *converges to the solution at time* s *of the ODE*

$$\begin{aligned} H(0) &= A_\infty x \\ \frac{dH}{ds}(s) &= f(H(s), \mathcal{Z}_\infty(s)), \quad s \in [0,1]. \end{aligned} \tag{3.45}$$

(iv) Uniformly over $x \in \mathcal{X}$, *the output* $F^L(x;t)$ *converges to* $F_\infty(x) = B_\infty H(1)$. *Furthermore,* $F_\infty(x_i) = y_i$ *for* $i \in \{1, \dots, n\}$.

Proof. The existence of limits A_∞ and B_∞ to $A^L(t)$ and $B^L(t)$ as L and t tend to infinity is given by Lemma 3.22. The same argument applies to $Z^L_{\lfloor sL \rfloor}(t)$, which provides a limit $\mathcal{Z}_\infty(s)$ to the sequence. Furthermore, following the proof of the lemma, we see that the convergence of $Z^L_{\lfloor sL \rfloor}(t)$ to $\mathcal{Z}_\infty(s)$ is uniform over $s \in [0,1]$. Corollary 3.14, applied with $\theta^L_k = Z^L_k$, $\Theta_\infty = \mathcal{Z}_\infty$, $a^L = A^L x$, $g = f$, then ensures that $h^L_{\lfloor Ls \rfloor}(t)$ converges uniformly (over $s \in [0,1]$ and $x \in \mathcal{X}$) to $H(s)$ that is the solution at time s of (3.45), as L and t tend to infinity. As a consequence, $F^L(x;t)$ converges uniformly over x to $F_\infty(x)$ as $L, t \to \infty$. Furthermore, recall that

$$\ell^L(t) = \frac{1}{n} \sum_{i=1}^n \|F^L(x_i;t) - y_i\|_2^2.$$

The left-hand side converges as $L, t \to \infty$ to 0 by assumption of the proposition, while the right-hand side converges to

$$\frac{1}{n} \sum_{i=1}^n \|F_\infty(x_i) - y_i\|_2^2.$$

Therefore, $F_\infty(x_i) = y_i$ for $i \in \{1, \dots, n\}$, and the proof is complete. □

3.B Proofs of the results of the main part of the chapter

Most of the results follow from those presented in Section 3.A. The only substantial proof is that of Proposition 3.6, which shows the local PL condition. It uses a result of Nguyen and Mondelli (2020) involving the Hermite transform and the sub-Gaussian variance proxy, which we define briefly. We refer to Debnath and Bhatta (2014, Chapter 17) and Vershynin (2018, Sections 2.5.2 and 3.4.1), respectively, for more detailed explanations.

Hermite transform. The r-th normalized probabilist's Hermite polynomial is given by

$$h_r(x) = \frac{1}{\sqrt{r!}}(-1)^r e^{x^2/2}\frac{d^r}{dx^r}e^{-x^2/2}, \quad r \geq 0.$$

This family of polynomials forms an orthonormal basis of square-integrable functions for the inner product

$$\langle f_1, f_2\rangle = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f_1(x)f_2(x)e^{-x^2/2}dx.$$

Therefore, any function σ such that $\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\sigma^2(x)e^{-x^2/2}dx < \infty$ can be decomposed on this basis. The r-th coefficient of this decomposition is denoted by $\eta_r(\sigma)$.

Sub-Gaussian random vector. A random vector $x \in \mathbb{R}^d$ is sub-Gaussian with variance proxy $v_x > 0$ if, for every $y \in \mathbb{R}^d$ of unit norm,

$$\mathbb{P}(|\langle x, y\rangle| \geq t) \leq 2\exp\Big(-\frac{t^2}{2v_x^2}\Big).$$

Additional notation. For a matrix A, we let $s_{\min}$ and $s_{\max}$ its minimum and maximum singular values, and similarly, $\lambda_{\min}$ and $\lambda_{\max}$ its minimum and maximum eigenvalues (whenever they exist).

Before delving into the proofs, we briefly describe the parts of this section that make use of the specific model (3.3). The most important one is the proof of Proposition 3.6, i.e., the proof that the residual network satisfies the (M, μ)-local PL condition. Additionally, in the proof of Proposition 3.3, the expressions for M and K are valid only for the specific model (3.3). Finally, in the proof of Theorem 3.7, the beginning of the proof reveals that condition (3.22) of Proposition 3.9 on μ can be expressed as a condition on the norm of the labels y_i. This applies only to the specific model (3.3). Observe that, if one assumes that the general residual network of Section 3.A satisfies the (M, μ)-local PL condition with μ given by (3.22), then the rest of the proof of Theorem 3.7 unfolds, and the conclusions of the theorem hold for the general model.

3.B.1 Proof of Proposition 3.1

Proposition 3.1 is a consequence of Proposition 3.8 with $f(h, (V, W)) = \frac{1}{\sqrt{m}}V\sigma(\frac{1}{\sqrt{q}}Wh)$.

3.B.2 Proof of Proposition 3.2

Proposition 3.12, with $\theta_k^L = (V_k^L, W_k^L)$, $\Theta = (\mathcal{V}, \mathcal{W})$, $a^L = Ax$, $g(h, (V, W)) = \frac{1}{\sqrt{m}}V\sigma(\frac{1}{\sqrt{q}}Wh)$, gives the existence and uniqueness of the solution of the neural ODE (3.6). Moreover, inspecting the proof of Proposition 3.12, equations (3.35) gives that, for any input $x \in \mathcal{X}$, the difference between the last hidden layer h_L^L of the discrete residual network (3.3) and its continuous counterpart $H(1)$ in the neural ODE (3.6) is bounded by

$$C'\Big(\frac{1}{L} + \sup_{s\in[0,1]}\|\Theta(s) - \Theta^L(s)\|\Big),$$

where $C' > 0$ is independent of L and $x \in \mathcal{X}$, and $\Theta^L(s) = \theta^L_{\lfloor(L-1)s\rfloor+1}$. The function Θ^L is a piecewise-constant interpolation of Θ with pieces of length $\frac{1}{L-1}$. Since Θ is Lipschitz continuous, the distance between Θ and Θ^L decreases as C''/L for some $C'' > 0$ depending on Θ but not on L. This yields $\|h_L^L - H(1)\| \leq \frac{C'(1++C'')}{L}$, where C' and C'' are independent of L and $x \in \mathcal{X}$. Since $F^L(x) = Bh_L^L$ and $F(x) = BH(1)$, the result is proven.

3.B.3 Proof of Proposition 3.3

We apply Proposition 3.8 with $f(h,(V,W)) = \frac{1}{\sqrt{m}} V\sigma(\frac{1}{\sqrt{q}} Wh)$. Recall that the parameters $Z = (V,W)$ are considered in Proposition 3.8 as a vector. In particular, $\|Z\| = \|V\|_F + \|W\|_F$. Therefore, Proposition 3.8 shows that, for $t \in [0,T]$, $L \in \mathbb{N}^*$, and $k \in \{1,\dots,L\}$,

$$\|A^L(t)\|_F \leq M, \quad \|V_k^L(t)\|_F + \|W_k^L(t)\|_F \leq M, \quad \text{and} \quad \|B^L(t)\|_F \leq M,$$

where

$$\begin{aligned} M &= M_0 + TM_\pi \\ M_0 &= \max\Big(\|A^L(0)\|_F,\, \|V_0^L(0)\|_F + \|W_0^L(0)\|_F,\, \|B^L(0)\|_F\Big) \\ M_\pi &= \max\Big(\max_{A\in\mathbb{R}^{q\times d}} \|\pi(A)\|_F,\, \max_{Z\in\mathbb{R}^{q\times m}\times\mathbb{R}^{m\times q}} \|\pi(Z)\|,\, \max_{B\in\mathbb{R}^{d'\times q}} \|\pi(B)\|_F\Big). \end{aligned}$$

Furthermore, due to our initialization scheme described in Section 3.3,

$$\|A^L(0)\|_F = \sqrt{d}, \quad \|V_0^L(0)\|_F = 0, \quad \|W_0^L(0)\|_F \leq 2\sqrt{qm}, \quad \|B^L(0)\|_F = \sqrt{d'},$$

where the third inequality holds with probability at least $1-\exp(-\frac{3qm}{16})$ by Lemma 3.23. Since we take $q \geq \max(d,d')$, this implies that, with high probability, $M_0 \leq 2\sqrt{qm}$, yielding the formula for M in Proposition 3.3. Finally, the existence of $K = \beta T e^{\alpha T}$ such that the difference between two successive weight matrices is bounded by K/L, as well as the dependence of α and β on $\mathcal{X}$, $\mathcal{Y}$, M, and σ, follows easily from Proposition 3.8, given that our initialization scheme ensures that $Z_k^L(0) = Z_{k+1}^L(0)$ for all $L \in \mathbb{N}^*$ and $k \in \{1,\dots,L\}$.

3.B.4 Proof of Theorem 3.4

By Proposition 3.3 and the fact that π is bounded, the sequences $(A^L)_{L\in\mathbb{N}^*}$ and $(B^L)_{L\in\mathbb{N}^*}$ each satisfy the assumptions of Corollary 3.11, and $(Z_k^L)_{L\in\mathbb{N}^*, 1\leq k\leq L}$ satisfies the assumptions of Proposition 3.10. Theorem 3.4 then follows directly from Theorem 3.15, by taking, as previously, $f(h,(V,W)) = \frac{1}{\sqrt{m}} V\sigma(\frac{1}{\sqrt{q}} Wh)$.

3.B.5 Proof of Proposition 3.6

We drop the L superscripts for this proof, since L is fixed. Denote by $\bar{A}, \bar{B}, \bar{V}_k, \bar{W}_k$ parameters sampled according to the initialization scheme of Section 3.3, which means in particular that $\bar{V}_k = 0$ and $\bar{W}_k = \bar{W} \sim \mathcal{N}^{\otimes(m\times q)}$. Since, by assumption, the activation function σ is bounded and not constant, it cannot be a polynomial function. As a consequence, there are infinitely many non-zero coefficients $\eta_r(\sigma)$ in its Hermite expansion (defined at the beginning of Section 3.B). Throughout, we let $r \geq 2$ be an integer such that $\eta_r(\sigma)$ is nonzero. We also let K_σ be the Lipschitz constant of σ and M_σ its supremum norm. Now, let A, B, V_k, W_k be parameters at distance at most $M = \min(\frac{\eta_r(\sigma)}{32K_\sigma\sqrt{2nq}}, \frac{1}{2})$ from $\bar{A}, \bar{B}, \bar{V}_k, \bar{W}_k$ in the sense of Definition 3.5.

It is useful for this proof to introduce a matrix-valued version of the residual network (3.3). More specifically, given data matrices $\mathbf{x} \in \mathbb{R}^{d\times n}$ and $\mathbf{y} \in \mathbb{R}^{d'\times n}$, the matrix-valued residual network writes

$$\begin{aligned} \mathbf{h}_0 &= A\mathbf{x} \\ \mathbf{h}_{k+1} &= \mathbf{h}_k + \frac{1}{L\sqrt{m}} V_{k+1}\sigma\Big(\frac{1}{\sqrt{q}} W_{k+1}\mathbf{h}_k\Big), \quad k \in \{0,\dots,L-1\}, \end{aligned} \tag{3.46}$$

where now $\mathbf{h}_k \in \mathbb{R}^{q\times n}$. The loss is equal to $\ell = \frac{1}{n}\|B\mathbf{h}_L - \mathbf{y}\|_F^2$ and we let $\mathbf{p}_k = \frac{\partial \ell}{\partial \mathbf{h}_k} \in \mathbb{R}^{q\times n}$ be the matrix-valued backward state. Observe that the columns of $\mathbf{x}$ are bounded and thus sub-Gaussian. In the sequel, we denote by v_x the sub-Gaussian variance proxy of the columns of $\sqrt{d/q}\mathbf{x}$.

Now that we have introduced the necessary notation, we can proceed to prove some preliminary estimates. Since $M \le \frac{1}{2} \le \sqrt{2qm}$, we have, for $k \in \{1,\dots,n\}$,

$$\|A - \bar{A}\|_F \le M, \quad \|B - \bar{B}\|_F \le \frac{1}{2}, \quad \|V_k\|_F \le 1, \quad \|W_k - \bar{W}\|_F \le \frac{1}{2} \le \sqrt{2qm}. \tag{3.47}$$

By Lemma 3.23, with probability at least $1 - \exp\big(-\frac{qm}{16}\big)$, one has $\|\bar{W}\|_F \le \sqrt{2qm}$. Together with the previous inequalities, this implies

$$\|A\|_2 \le 2, \quad s_{\min}(B) \ge \frac{1}{2}, \quad \|B\|_2 \le \frac{3}{2}, \quad \|V_k\|_F \le 1, \quad \|W_k\|_F \le 2\sqrt{2qm}, \tag{3.48}$$

where the second inequality is a consequence of Lemma 3.21, as follows:

$$s_{\min}(B) \ge s_{\min}(\bar{B}) - \|B - \bar{B}\|_F = 1 - \|B - \bar{B}\|_F \ge \frac{1}{2}.$$

Let us now bound $\|\mathbf{h}_k\|_F$ and $\|\mathbf{p}_k\|_F$. We have

$$\|\mathbf{h}_0\|_F = \|A\mathbf{x}\|_F \le \|A\|_2\|\mathbf{x}\|_F \le 2\sqrt{qn}. \tag{3.49}$$

Moreover, by (3.46), for any $k \in \{0,\dots,L-1\}$,

$$\|\mathbf{h}_{k+1}\|_F \le \|\mathbf{h}_k\|_F + \frac{K_\sigma}{L\sqrt{m}\sqrt{q}}\|V_{k+1}\|_F\|W_{k+1}\|_F\|\mathbf{h}_k\|_F \le \Big(1 + \frac{2\sqrt{2}K_\sigma}{L}\Big)\|\mathbf{h}_k\|_F,$$

where the second inequality is a consequence of (3.48). Therefore, by (3.49),

$$\|\mathbf{h}_k\|_F \le \exp(2\sqrt{2}K_\sigma)\|\mathbf{h}_0\|_F \le 2\exp(2\sqrt{2}K_\sigma)\sqrt{qn}. \tag{3.50}$$

Moving on to $\|\mathbf{p}_k\|_F$, the chain rule leads to

$$\mathbf{p}_k = \mathbf{p}_{k+1} + \frac{1}{L\sqrt{qm}}W_{k+1}^\top\Big((V_{k+1}^\top\mathbf{p}_{k+1}) \odot \sigma'\big(\frac{1}{\sqrt{q}}W_{k+1}\mathbf{h}_k\big)\Big), \quad k \in \{0,\dots,L-1\},$$

where $\odot$ denotes the element-wise product. Noting that $|\sigma'| \le K_\sigma$ and using (3.48), we obtain

$$\|\mathbf{p}_k\|_F \ge \|\mathbf{p}_{k+1}\|_F - \frac{K_\sigma}{L\sqrt{qm}}\|W_{k+1}\|_F\|V_{k+1}\|_F\|\mathbf{p}_{k+1}\|_F \ge \Big(1 - \frac{2\sqrt{2}K_\sigma}{L}\Big)\|\mathbf{p}_{k+1}\|_F.$$

It follows that $\|\mathbf{p}_k\|_F \ge \exp(-2\sqrt{2}K_\sigma)\|\mathbf{p}_L\|_F$. In addition,

$$\mathbf{p}_L = \frac{\partial \ell}{\partial \mathbf{h}_L} = \frac{2}{n}B^\top(B\mathbf{h}_L - \mathbf{y}).$$

Therefore, by Lemma 3.21, since $d' \le q$,

$$\|\mathbf{p}_L\|_F \ge \frac{2}{n}s_{\min}(B)\|B\mathbf{h}_L - \mathbf{y}\|_F \ge \frac{1}{\sqrt{n}}\sqrt{\ell}.$$

Collecting bounds, we conclude that, for $k \in \{0, \dots, L\}$,

$$\|\mathbf{p}_k\|_F \geq \frac{1}{\sqrt{n}} \exp(-2\sqrt{2}K_\sigma)\sqrt{\ell}. \tag{3.51}$$

A similar proof reveals that, for $k \in \{0, \dots, L\}$,

$$\|\mathbf{p}_k\|_F \leq \frac{3}{\sqrt{n}} \exp(2\sqrt{2}K_\sigma)\sqrt{\ell}.$$

Having established these preliminary estimates, our goal in the remainder of the proof is to lower bound the quantity $\|\frac{\partial \ell}{\partial V_{k+1}}\|_F$. First note that, by the chain rule, for any $k \in \{0, \dots, L-1\}$,

$$\frac{\partial \ell}{\partial V_{k+1}} = \frac{1}{L\sqrt{m}} \mathbf{p}_{k+1} \sigma\Big(\frac{1}{\sqrt{q}} W_{k+1}\mathbf{h}_k\Big)^\top.$$

As a consequence, when $m \geq n$, by Lemma 3.21,

$$\begin{aligned}\Big\|\frac{\partial \ell}{\partial V_{k+1}}\Big\|_F &\geq \frac{1}{L\sqrt{m}} \|\mathbf{p}_{k+1}\|_F \cdot s_{\min}\Big(\sigma\Big(\frac{1}{\sqrt{q}} W_{k+1}\mathbf{h}_k\Big)\Big) \\ &\geq \frac{1}{L\sqrt{mn}} \exp(-2\sqrt{2}K_\sigma)\sqrt{\ell} \cdot s_{\min}\Big(\sigma\Big(\frac{1}{\sqrt{q}} W_{k+1}\mathbf{h}_k\Big)\Big), \end{aligned} \tag{3.52}$$

using (3.51). Next, by Lemma 3.21,

$$s_{\min}\Big(\sigma\Big(\frac{1}{\sqrt{q}} W_{k+1}\mathbf{h}_k\Big)\Big) \geq s_{\min}\Big(\sigma\Big(\frac{1}{\sqrt{q}} \bar{W}\bar{A}\mathbf{x}\Big)\Big) - \Big\|\sigma\Big(\frac{1}{\sqrt{q}} W_{k+1}\mathbf{h}_k\Big) - \sigma\Big(\frac{1}{\sqrt{q}} \bar{W}\bar{A}\mathbf{x}\Big)\Big\|_F.$$

Let us first lower bound the first term. Since, by our choice of initialization, $\bar{A} = (I_{\mathbb{R}^{d\times d}}, 0_{\mathbb{R}^{(q-d)\times d}})$, we have

$$s_{\min}\Big(\sigma\Big(\frac{1}{\sqrt{q}} \bar{W}\bar{A}\mathbf{x}\Big)\Big) = s_{\min}(\sigma(\tilde{W}\tilde{\mathbf{x}})),$$

where $\tilde{W} \sim \mathcal{N}(0,1)^{\otimes(m\times d)}$ and $\tilde{\mathbf{x}} = \frac{1}{\sqrt{q}}\mathbf{x} \in \mathbb{R}^{d\times n}$ has i.i.d. unitary columns independent of $\tilde{W}$. Therefore, by Lemma 3.24, with probability at least $1 - \exp\big(-\frac{3m\eta_r^2(\sigma)}{64M_\sigma^2 n}\big) - 2n^2 \exp\big(-\frac{d}{2v_x n^{2/r}}\big)$,

$$s_{\min}\Big(\sigma\Big(\frac{1}{\sqrt{q}} W\bar{A}\mathbf{x}\Big)\Big) \geq \frac{\sqrt{m}\eta_r(\sigma)}{4}.$$

Next,

$$\begin{aligned}\Big\|\sigma\Big(\frac{1}{\sqrt{q}} W_{k+1}\mathbf{h}_k\Big) - \sigma\Big(\frac{1}{\sqrt{q}} \bar{W}\bar{A}\mathbf{x}\Big)\Big\|_F \leq \frac{K_\sigma}{\sqrt{q}}\Big(&\|W_{k+1} - \bar{W}\|_F\|\mathbf{h}_k\|_F + \|\bar{W}\|_F\|\mathbf{h}_k - A\mathbf{x}\|_F \\ &+ \|\bar{W}\|_F\|A\mathbf{x} - \bar{A}\mathbf{x}\|_F\Big).\end{aligned}$$

Clearly,

$$\|\mathbf{h}_k - A\mathbf{x}\|_F = \Big\|\sum_{j=1}^{k} \frac{1}{L\sqrt{m}} V_j \sigma\Big(\frac{1}{\sqrt{q}} W_j \mathbf{h}_{j-1}\Big)\Big\|_F \leq \frac{4\sqrt{2}K_\sigma k}{L} \exp(2\sqrt{2}K_\sigma)\sqrt{qn},$$

by (3.48) and (3.50). Also,

$$\|A\mathbf{x} - \bar{A}\mathbf{x}\|_F \leq \|A - \bar{A}\|_F\|\mathbf{x}\|_F \leq \frac{\eta_r(\sigma)}{32\sqrt{2}K_\sigma},$$

by (3.47) and by definition of M. Putting together the two bounds above as well as (3.47), (3.48), and (3.50), we obtain

$$\begin{aligned}\Big\|\sigma\Big(\frac{1}{\sqrt{q}}W_{k+1}\mathbf{h}_k\Big)-\sigma\Big(\frac{1}{\sqrt{q}}W\bar{A}\mathbf{x}\Big)\Big\|_F &\le K_\sigma\exp(2\sqrt{2}K_\sigma)\sqrt{n}\Big(1+\sqrt{qm}\frac{8K_\sigma k}{L}\Big)+\sqrt{m}\frac{\eta_r(\sigma)}{32}\\ &\le C_1\sqrt{n}+C_2\frac{\sqrt{nqm}k}{16L}+\sqrt{m}\frac{\eta_r(\sigma)}{32},\end{aligned}$$

where $C_1 = K_\sigma\exp(2\sqrt{2}K_\sigma)$ and $C_2 = 128C_1K_\sigma$. Thus, when $C_1\sqrt{n} \le \frac{1}{32}\sqrt{m}\eta_r(\sigma)$, we have

$$s_{\min}\Big(\sigma\Big(\frac{1}{\sqrt{q}}W_{k+1}\mathbf{h}_k\Big)\Big) \ge \sqrt{m}\Big(\frac{3}{16}\eta_r(\sigma)-\frac{C_2}{16}\sqrt{nq}\frac{k}{L}\Big) \ge \frac{1}{8}\sqrt{m}\eta_r(\sigma)$$

for $k \le \frac{L\eta_r(\sigma)}{C_2\sqrt{nq}}$. As a consequence, for $k \le \frac{L\eta_r(\sigma)}{C_2\sqrt{nq}}$, returning to (3.52),

$$\Big\|\frac{\partial\ell}{\partial V_{k+1}}\Big\|_F \ge \frac{1}{8L\sqrt{n}}\eta_r(\sigma)\exp(-2\sqrt{2}K_\sigma)\sqrt{\ell} = \frac{C_3\eta_r(\sigma)}{L\sqrt{n}}\sqrt{\ell},$$

letting $C_3 = \frac{\exp(-2\sqrt{2}K_\sigma)}{8}$. Therefore,

$$\begin{aligned}\Big\|\frac{\partial\ell}{\partial A}\Big\|_F^2+L\sum_{k=1}^{L}\Big\|\frac{\partial\ell}{\partial Z_{k+1}}\Big\|_F^2+\Big\|\frac{\partial\ell}{\partial B}\Big\|_F^2 &\ge L\sum_{k=1}^{\lfloor\frac{L\eta_r(\sigma)}{C_2\sqrt{nq}}\rfloor}\Big\|\frac{\partial\ell}{\partial V_{k+1}}\Big\|_F^2\\ &\ge L\Big\lfloor\frac{L\eta_r(\sigma)}{C_2\sqrt{nq}}\Big\rfloor\frac{C_3^2\eta_r(\sigma)^2}{L^2n}\ell\\ &\ge \frac{C_3^2\eta_r(\sigma)^3}{2C_2n\sqrt{nq}}\ell,\end{aligned}$$

where we used the inequality $\lfloor x\rfloor \ge x/2$ for $x \ge 1$. This proves the result, with

$$\begin{aligned}c_1 &= \max\Big(\frac{2^{10}C_1^2}{\eta_r(\sigma)^2},1\Big) = \max\Big(\frac{2^{10}K_\sigma^2\exp(4\sqrt{2}K_\sigma)}{\eta_r(\sigma)^2},1\Big)\\ c_2 &= \frac{C_2}{\eta_r(\sigma)} = \frac{128K_\sigma^2\exp(2\sqrt{2}K_\sigma)}{\eta_r(\sigma)}\\ c_3 &= \min\Big(\frac{\eta_r(\sigma)}{32\sqrt{2}K_\sigma},\frac{1}{2}\Big)\\ c_4 &= \frac{C_3^2\eta_r(\sigma)^3}{2C_2} = \frac{\eta_r(\sigma)^3}{2^{14}K_\sigma^2\exp(6\sqrt{2}K_\sigma)}\\ \delta &= \exp\big(-\frac{qm}{16}\big)+n\exp\Big(-\frac{3m\eta_r^2(\sigma)}{64M_\sigma^2n}\Big)+2n^2\exp\Big(-\frac{d}{2v_xn^{2/r}}\Big).\end{aligned}$$

Remark 3.17. *With appropriate values of r and m, the probability of failure δ can be made as small as*

$$\varepsilon+2n^2\exp\Big(-\frac{d}{2v_xn^{\varepsilon}}\Big), \tag{3.53}$$

for any $\varepsilon > 0$. This is possible first by choosing r such that $2/r \ge \varepsilon$, then by choosing m such that the first two terms are less than ε. Moreover, we refer the interested reader to Goel et al. (2020, Lemmas A.2 and A.9) for quantitative estimates of $\eta_r(\sigma)$ for ReLU and sigmoid activations. Finally, the expression (3.53) is essentially the same as the one appearing in Nguyen and Mondelli (2020, Theorem 3.3). As in this chapter, we note that this expression is small if n grows at most polynomially with d, in which case the exponential term in d dominates the polynomial term in n.

3.B.6 Proof of Theorem 3.7

By Proposition 3.6, there exists $\delta > 0$ such that, with probability at least $1 - \delta$, the residual network (3.3) satisfies the (M, μ)-local PL condition around its initialization, with

$$M = \frac{c_3}{\sqrt{nq}} \quad \text{and} \quad \mu = \frac{c_4}{n\sqrt{nq}},$$

for c_3 and c_4 depending on σ. We now apply Proposition 3.9 with $f(h, (V, W)) = \frac{1}{\sqrt{m}} V \sigma(\frac{1}{\sqrt{q}} W h)$. The only assumption of Proposition 3.9 that requires some care to check is that the PL condition holds for the value of μ given by equation (3.22). Since the (M, μ)-local PL condition implies the $(M, \tilde{\mu})$-local PL condition for any $\tilde{\mu} \in (0, \mu)$, it is the case if

$$\frac{c_4}{n\sqrt{nq}} \geq \max(M_B K, M_B M_X, M_A M_X) \frac{8e^K}{M} \sup_{L \in \mathbb{N}^*} \sqrt{\ell^L(0)},$$

with M_X, M_A, M_B, and K defined in Proposition 3.9. Due to the initialization scheme of Section 3.3, we have, for any input $x \in \mathcal{X}$, $h_L^L(0) = h_0^L(0)$, hence $F^L(x) = B^L(0)A^L(0)x = 0$ since $q \geq d + d'$. As a consequence, $\ell^L(0) = \frac{1}{n} \sum_{i=1}^n \|y_i\|^2$. Therefore, the condition becomes

$$\frac{1}{n} \sum_{i=1}^n \|y_i\|^2 \leq \frac{c_3^2 c_4^2}{64 n^4 q^3 \max(M_B K, M_B M_X, M_A M_X)^2 e^{2K}},$$

where we replaced M by its value. Define C to be equal to the constant on the right-hand side. Then, according to the above, as soon as $\frac{1}{n} \sum_{i=1}^n \|y_i\|^2 \leq C$, we can apply Proposition 3.9, which gives several guarantees. First, the gradient flow is well defined on $\mathbb{R}_+$. Moreover, the proposition and the expression of μ given above yield the bound on the empirical risk. In particular, the empirical risk converges uniformly to zero. Furthermore, Proposition 3.9 shows the uniform convergence of the weights as $t \to \infty$. Finally, the proposition ensures that the sequences $(A^L)_{L \in \mathbb{N}^*}$ and $(B^L)_{L \in \mathbb{N}^*}$ each satisfy the assumptions of Corollary 3.11, and that $(Z_k^L)_{L \in \mathbb{N}^*, 1 \leq k \leq L}$ satisfies the assumptions of Proposition 3.10. We can therefore apply Theorem 3.15, with f defined above and π equal to the identity. This gives the uniform convergence of the weights as $L \to \infty$. The four asymptotic statements of Theorem 3.7 are then a consequence of Proposition 3.16.

Remark 3.18. *A close examination of the quantities involved in the definition of C reveals that it depends only on $\mathcal{X}$, σ, n, and q. In particular, it does not depend on the dimension m.*

3.C Some technical lemmas

We start by recalling the Picard-Lindelöf theorem (see, e.g., Luk, 2017, for a self-contained presentation, and Arnold, 1992, for a textbook).

Lemma 3.19 (Picard-Lindelöf theorem)**.** *Let $I = [0, T] \subset \mathbb{R}_+$ be an interval, for some $T \in (0, \infty]$. Consider the initial value problem*

$$U(s) = U_0 + \int_0^s g(U(r), r) dr, \quad s \in I, \tag{3.54}$$

where $g : \mathbb{R}^d \times I \to \mathbb{R}^d$ is continuous and locally Lipschitz continuous in its first variable. Then the initial value problem is well defined on an interval $[0, T_{\max}) \subset I$, i.e., there exists a unique maximal solution on this interval. Moreover, if $T_{\max} < T$, then $\|U(s)\|$ tends to infinity when s tends to $T_{\max}$. Finally, if $g(\cdot, r)$ is uniformly Lipschitz continuous for r in any compact, then $T_{\max} = T$.

We define time-dependent dynamics (3.54) for generality, but the time-independent case $U(s) = U_0 + \int_0^s g(U(r))dr$ is also of interest. In this case, the existence and uniqueness of the maximal solution holds if g is locally Lipschitz continuous, and the solution is defined on I if g is Lipschitz continuous. Besides, the first statement of Lemma 3.19 (existence and uniqueness of the maximal solution) also holds if $\mathbb{R}^d$ is replaced by any (potentially infinite-dimensional) Banach space.

The next lemma gives conditions for the existence and uniqueness of the global solution of the initial value problem (3.54) when the assumption of continuity of g in its second variable is removed, thereby generalizing the Picard-Lindelöf theorem.

Lemma 3.20 (Caratheodory conditions for the existence and uniqueness of the global solution of an initial value problem)**.** *Consider the initial value problem*

$$U(s) = U_0 + \int_0^s g(U(r), r)dr, \quad s \in [0, 1],$$

where $g : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ is measurable and the integral is understood in the sense of Lebesgue integration. Assume that $g(\cdot, r)$ is uniformly Lipschitz continuous for almost all $r \in [0, 1]$, and that $g(0, r) \equiv 0$. Then there exists a unique solution to the initial value problem, defined on $[0, 1]$.

Proof. The proof is a consequence of Filippov (1988, Theorems 1, 2, and 4). More specifically, denote by $C > 0$ the uniform Lipschitz constant of $g(\cdot, r)$. According to Filippov (1988, Theorems 1 and 2), under the conditions of the lemma, there exists a unique maximal solution to the initial value problem. Let us now consider a restricted version of the problem, where g is defined on $D \times [0, 1]$, with D a compact of $\mathbb{R}^d$ large enough to contain in its interior the ball of center 0 and radius $\|U_0\| \exp(C)$. There exists a unique maximal solution to this problem as well, also according to Filippov (1988, Theorems 1 and 2), and, according to Filippov (1988, Theorem 4), it is defined until it reaches the boundary of $D \times [0, 1]$, which it reaches at some point (U^*, s^*). If $s^* < 1$, it means that U^* is on the boundary of D, and in particular that $\|U^*\| > \|U_0\| \exp(C)$. But, on the other hand, for almost every $r \in [0, 1]$,

$$\|g(U(r), r)\| \leq \|g(0, r)\| + \|g(U(r), r) - g(0, r)\| \leq C\|U(r)\|.$$

Hence, by Grönwall's inequality, for $s \leq s^*$,

$$\|U(s)\| \leq \|U_0\| \exp(C).$$

Thus, $\|U^*\| \leq \|U_0\| \exp(C)$, which is impossible. Hence the maximal solution of the restricted problem is defined on $[0, 1]$. Furthermore, the maximal solution of the original problem coincides with the restricted one whenever $U(s) \in D$, which is the case for every $s \in [0, 1]$, hence the maximal solution is defined on $[0, 1]$. □

The next three lemmas recall well-known results from linear algebra, analysis, and random matrix theory. Recall that $s_{\min}$ and $\lambda_{\min}$ denote respectively the minimum singular value and eigenvalue of a matrix.

Lemma 3.21. *Let $A, A' \in \mathbb{R}^{m \times r}$ and $B \in \mathbb{R}^{r \times n}$. Then*

$$s_{\min}(A + A') \geq s_{\min}(A) - \|A'\|_F.$$

If $m \geq r$, then $\|AB\|_F \geq s_{\min}(A)\|B\|_F$. Furthermore, if $n \geq r$, then $\|AB\|_F \geq \|A\|_F s_{\min}(B)$.

Proof. The first statement is a consequence of, e.g., Loyka (2015), which establishes that $s_{\min}(A + A') \geq s_{\min}(A) - s_{\max}(A')$, yielding the first inequality since $s_{\max}(A') = \|A\|_2 \leq \|A\|_F$. As for the second one, we have

$$\|AB\|_F^2 = \mathrm{Tr}(ABB^\top A^\top) = \mathrm{Tr}(BB^\top A^\top A) \geq \lambda_{\min}(A^\top A)\mathrm{Tr}(BB^\top) = \lambda_{\min}(A^\top A)\|B\|_F^2.$$

Since $m \geq r$, the rightmost quantity is equal to $s_{\min}(A)\|B\|_F$, proving the second statement of the lemma. The third statement is similar. □

Lemma 3.22. *Let $(e_{x,y})_{x\in\mathbb{R}_+,y\in\mathbb{R}_+} \subset E$, where E is a Banach space, such that $e_{x,y}$ converges uniformly to $e_{\infty,y}$ when $x \to \infty$, and converges uniformly to $e_{x,\infty}$ when $y \to \infty$. Then there exists $e_\infty \in E$ such that*

$$\lim_{x,y\to\infty} e_{x,y} = \lim_{x\to\infty} e_{x,\infty} = \lim_{y\to\infty} e_{\infty,y} = e_\infty.$$

Proof. Let $\varepsilon > 0$. Since $e_{x,y}$ converges uniformly to $e_{\infty,y}$ as $x \to \infty$, there exists $x_0 \in \mathbb{R}_+$ such that, for $x_1, x_2 > x_0$ and $y \in \mathbb{R}_+$,

$$\|e_{x_1,y} - e_{x_2,y}\| \leq \frac{\varepsilon}{2}.$$

Similarly, there exists $y_0 \in \mathbb{R}_+$ such that, for $x \in \mathbb{R}_+$ and $y_1, y_2 > y_0$,

$$\|e_{x,y_1} - e_{x,y_2}\| \leq \frac{\varepsilon}{2}.$$

Hence, for $x_1, x_2 > x_0$ and $y_1, y_2 > y_0$,

$$\|e_{x_1,y_1} - e_{x_2,y_2}\| \leq \|e_{x_1,y_1} - e_{x_1,y_2}\| + \|e_{x_1,y_2} - e_{x_2,y_2}\| \leq \varepsilon.$$

We conclude that $(e_{x,y})_{x\in\mathbb{R}_+,y\in\mathbb{R}_+}$ is a Cauchy sequence, which therefore converges to some limit $e_\infty \in E$. □

Lemma 3.23. *Let $W \in \mathbb{R}^{q\times m}$ be a standard Gaussian random matrix. Then, for $M_W \geqslant \sqrt{2}$, with probability at least $1 - \exp(-\frac{(M_W^2-1)qm}{16})$, one has $\|W\|_F \leq M_W\sqrt{q}\sqrt{m}$.*

Proof. The quantity $\|W\|_F^2$ follows a chi-squared distribution with qm degrees of freedom. Hence, according to Laurent and Massart (2000, Lemma 1), for $x \geq 0$,

$$\mathbb{P}(\|W\|_F^2 - qm \geq 2\sqrt{qmx} + 2x) \leq \exp(-x).$$

Taking $x = \frac{(M_W^2-1)qm}{16}$, we see that

$$2\sqrt{qmx} = \frac{1}{2}\sqrt{M_W^2 - 1}qm \leq \frac{1}{2}(M_W^2 - 1)qm,$$

where the bound follows from $M_W \geqslant \sqrt{2}$. Since furthermore $2x \leq \frac{1}{2}(M_W^2 - 1)qm$, we obtain

$$2\sqrt{qmx} + 2x \leq (M_W^2 - 1)qm,$$

and thus

$$\mathbb{P}(\|W\|_F^2 > M_W^2 qm) \leq \mathbb{P}(\|W\|_F^2 - qm \geq 2\sqrt{qmx} + 2x) \leq \exp(-x),$$

yielding the result. □

Finally, the last lemma of the section gives a lower bound on the smallest singular value of a matrix of the form $\sigma(A)$, where σ is a bounded function applied element-wise and A belongs to a family of random matrix. The lower bound involves the Hermite transform of σ, which is defined in Section 3.B.

Lemma 3.24. *Let σ be a function bounded by some $M_\sigma > 0$. Let $W \in \mathbb{R}^{m\times d}$ be a standard Gaussian random matrix, and $X \in \mathbb{R}^{d\times n}$ a random matrix with i.i.d. unitary columns independent of W. Then, for any integer $r \geq 2$, there exists $\delta > 0$ such that, with probability at least $1-\delta$, the smallest singular value of $\sigma(WX)$ is greater than $\frac{1}{4}\sqrt{m}\eta_r(\sigma)$, where $\eta_r(\sigma)$ is the r-th coefficient in the Hermite transform of σ. Furthermore, the following expression for δ holds:*

$$\delta = n\exp\Big(-\frac{3m\eta_r^2(\sigma)}{64M_\sigma^2 n}\Big) + 2n^2\exp\Big(-\frac{d}{2Cn^{2/r}}\Big),$$

where C is the sub-Gaussian variance proxy of the columns of $\sqrt{d}X$.

Proof. Denoting by w_i the i-th row of W and letting

$$M_i = \sigma(X^\top w_i^\top)\sigma(w_i X),$$

our goal is to lower bound the smallest eigenvalue value $\lambda_{\min}(M)$ of $M = \sum_{i=1}^m M_i$. Observe that

$$\begin{aligned}\mathbb{E}(M|X) &= m\mathbb{E}_{\tilde{w}\sim\mathcal{N}(0,I_d)}\Big(\sigma(X^\top\tilde{w}^\top)\sigma(\tilde{w}X)\Big|X\Big)\\ &= m\mathbb{E}_{\tilde{w}\sim\mathcal{N}(0,\frac{1}{d}I_d)}\Big(\sigma\big((\sqrt{d}X)^\top\tilde{w}^\top\big)\sigma\big(\tilde{w}(\sqrt{d}X)\big)\Big|X\Big).\end{aligned}$$

Letting $\lambda_{\min}(\mathbb{E}(M|X))$ be the smallest eigenvalue of this matrix and $r \geq 2$ be an integer, Nguyen and Mondelli (2020, Lemma 3.4) show that, with probability at least $1 - 2n^2\exp(-\frac{d}{2Cn^{2/r}})$ over the matrix X,

$$\lambda_{\min}(\mathbb{E}(M|X)) \geq \frac{m\eta_r^2(\sigma)}{8}. \tag{3.55}$$

We now apply a matrix Chernoff's bound to lower bound with high probability the smallest eigenvalue $\lambda_{\min}(M|X)$ of M conditionally on X, as a function of $\lambda_{\min}(\mathbb{E}(M|X))$. By Tropp (2012, Remark 5.3), we have, for $t \in [0,1]$,

$$\mathbb{P}(\lambda_{\min}(M) \leq t\lambda_{\min}(\mathbb{E}(M|X))|X) \leq n\exp\Big(-\frac{(1-t^2)\lambda_{\min}(\mathbb{E}(M|X))}{2R(X)}\Big),$$

where $R(X)$ is an almost sure upper bound on the largest eigenvalue of $M_i|X$, which we can take equal to $M_\sigma^2 n$ since the largest eigenvalue of M_i is equal to $\|\sigma(w_i X)\|_2^2 \leq M_\sigma^2 n$. Taking $t = 1/2$, we obtain, on the event $[\lambda_{\min}(\mathbb{E}(M|X)) \geq \frac{m\eta_r^2(\sigma)}{8}]$,

$$\mathbb{P}\Big(\lambda_{\min}(M) \geq \frac{\lambda_{\min}(\mathbb{E}(M|X))}{2}\Big|X\Big) \geq 1 - n\exp\Big(-\frac{3m\eta_r^2(\sigma)}{64M_\sigma^2 n}\Big),$$

thus, on the event $[\lambda_{\min}(\mathbb{E}(M|X)) \geq \frac{m\eta_r^2(\sigma)}{8}]$,

$$\mathbb{P}\Big(\lambda_{\min}(M) \geq \frac{m\eta_r^2(\sigma)}{16}\Big) \geq 1 - n\exp\Big(-\frac{3m\eta_r^2(\sigma)}{64M_\sigma^2 n}\Big).$$

Using (3.55), we obtain

$$\begin{aligned}\mathbb{P}\Big(\lambda_{\min}(M) \geq \frac{m\eta_r^2(\sigma)}{16}\Big) &\geq \Big(1 - n\exp\Big(-\frac{3m\eta_r^2(\sigma)}{64M_\sigma^2 n}\Big)\Big)\mathbb{P}\Big(\lambda_{\min}(\mathbb{E}(M|X)) \geq \frac{m\eta_r^2(\sigma)}{8}\Big) \\ &\geq \Big(1 - n\exp\Big(-\frac{3m\eta_r^2(\sigma)}{64M_\sigma^2 n}\Big)\Big)\Big(1 - 2n^2\exp\Big(-\frac{d}{Cn^{2/r}}\Big)\Big) \\ &\geq 1 - n\exp\Big(-\frac{3m\eta_r^2(\sigma)}{64M_\sigma^2 n}\Big) - 2n^2\exp\Big(-\frac{d}{2Cn^{2/r}}\Big).\end{aligned}$$

□

3.D Counter-example for the ReLU case.

This section gives a proof sketch to illustrate that, with the ReLU activation $\sigma : x \mapsto \max(0, x)$, the smoothness of the weights can be lost during training. More precisely, we show a case where successive weights are at distance $\mathcal{O}(\frac{1}{L})$ at initialization and at distance $\Omega(1)$ after training.

For the sake of simplicity, we will assume that the depth is even, and denote it as $2L$. We place ourselves in a one-dimensional setting (i.e., $d = 1$). The parameters are $(w_1, \cdots, w_{2L}) \in \mathbb{R}^{2L}$, and the residual network writes as follows, for an input $x \in \mathbb{R}$:

$$\begin{aligned}h_0(t) &= x \\ h_{k+1}(t) &= h_k(t) + \frac{1}{2L}\sigma(w_{k+1}(t)h_k(t)), \quad k \in \{0, \dots, 2L-1\}.\end{aligned}$$

We consider a sample consisting of a single point $(x, Cx) \in \mathbb{R}_+^2$, with $C > 1$ (independent of L), and define the empirical risk as $\ell(t) = (h_{2L}(t) - Cx)^2$. The risk is minimized by gradient flow.

The weights are initialized to $w_k(0) = \frac{(-1)^k}{2L}$. For $x \in \mathbb{R}_+$ we have that $h_k(t) \geq 0$ for all $k \in \{0, \dots, 2L\}$. Note that the argument of σ on the odd layers is negative. Therefore, by definition of σ, the gradient of the loss with respect to the odd layers is zero and we have, for $k \in \{0, \dots, L-1\}$, $w_{2k+1}(t) = w_{2k+1}(0)$. On the other hand, the argument of σ is positive on the even layers, and thus,

$$h_{2L}(t) = \prod_{j=1}^{L}\Big(1 + \frac{w_{2j}(t)}{2L}\Big)x.$$

As a consequence, the gradient flow equation for the even layers is, for $k \in \{1, \dots, L\}$,

$$\frac{dw_{2k}}{dt}(t) = -\frac{\partial \ell}{\partial w_{2k}}(t) = 2x\Big(C - \prod_{j=1}^{L}\Big(1 + \frac{w_{2j}(t)}{2L}\Big)\Big)\prod_{j=1, j\neq k}^{L}\Big(1 + \frac{w_{2j}(t)}{2L}\Big).$$

Due to the symmetry of these equations for $k \in \{1, \dots, L\}$ and the fact that all the $w_{2k}(0)$ are equal, the parameters on each even layer coincide at all times and are equal to $w(t)$ such that

$$\frac{dw}{dt}(t) = 2x\Big(C - \Big(1 + \frac{w(t)}{2L}\Big)^L\Big)\Big(1 + \frac{w(t)}{2L}\Big)^{L-1}.$$

An analysis of this ODE reveals that $w(t)$ tends as $t \to \infty$ to $w^\star > 0$ satisfying that

$$\Big(1 + \frac{w^\star}{2L}\Big)^L = C. \tag{3.56}$$

This can be seen by letting $y(t) = C - (1 + \frac{w(t)}{2L})^L$, and applying Grönwall's inequality to y. Therefore, as $t \to \infty$, one has $w_{2k+1}(t) \to -\frac{1}{2L}$ and $w_{2k}(t) \to w^\star$, where (3.56) implies that $w^\star \geq 2\log(C)$. This shows that the final weights are not smooth in the sense that the distance between two successive weights is $\Omega(1)$.

This result contrasts sharply with Proposition 3.8, which shows that successive weights remain at a distance $\mathcal{O}(\frac{1}{L})$ throughout training, when initialized as a discretization of a Lipschitz continuous function, and with a smooth activation function. In fact, Proposition 3.8 can be generalized to any initialization such that successive weights are at distance $\mathcal{O}(\frac{1}{L})$ at initialization, which is the case in the counter-example. This means that the only broken assumption in our counter-example is the non-smoothness of the activation function. This non-smoothness causes the gradient flow dynamics for two successive weights to deviate, even though the weights are initially close to each other, because they are separated by the kink of ReLU at zero.

3.E Experimental details

We use PyTorch (Paszke et al., 2019).

Large-depth limit. We take $n = 100$, $d = 16$, $m = 32$. We train for 500 iterations, and set the learning rate to $L \times 10^{-2}$. The scaling of the learning rate with L is the equivalent of the L factor in the gradient flow (3.4).

Long-time limit. We take $n = 50$, $d = 16$, $m = 64$, $L = 64$, and train for 80,000 iterations with a learning rate of $5L \times 10^{-3}$.

Real-world data. We take $L = 256$. The first layer is a trainable convolutional layer with a kernel size of 5×5, a stride of 2, a padding of 1, and 16 out channels. We then iterate the residual layers

$$h^L_{k+1} = h^L_k + \frac{1}{L}\mathrm{bn}^L_{2,k}(\mathrm{conv}^L_{2,k}(\sigma(\mathrm{bn}^L_{1,k}(\mathrm{conv}^L_{1,k}(h^L_k))))), \quad k \in \{0, \dots, L-1\},$$

where $\mathrm{conv}^L_{i,k}$ are convolutions with kernel size 3, stride of 2, and padding of 1, and $\mathrm{bn}^L_{i,k}$ are batch normalizations, as is standard in residual networks (He et al., 2016a). The model is trained using stochastic gradient descent on the cross-entropy loss for 180 epochs. The initial learning rate is 4×10^{-2} and is gradually decreased using a cosine learning rate scheduler.

Part II

Analogy between Residual Neural Networks and Neural Ordinary Differential Equations to design and study new architectures

4 Momentum Residual Neural Networks

The training of deep residual neural networks (ResNets) with backpropagation has a memory cost that increases linearly with respect to the depth of the network. A way to circumvent this issue is to use reversible architectures. In this chapter, we propose to change the forward rule of a ResNet by adding a momentum term. The resulting networks, momentum residual neural networks (Momentum ResNets), are invertible. Unlike previous invertible architectures, they can be used as a drop-in replacement for any existing ResNet block. We show that Momentum ResNets can be interpreted in the infinitesimal step size regime as second-order ordinary differential equations (ODEs) and exactly characterize how adding momentum progressively increases the representation capabilities of Momentum ResNets: they can learn any linear mapping up to a multiplicative factor, while ResNets cannot. In a learning to optimize setting, where convergence to a fixed point is required, we show theoretically and empirically that our method succeeds while existing invertible architectures fail. We show on CIFAR and ImageNet that Momentum ResNets have the same accuracy as ResNets, while having a much smaller memory footprint, and show that pre-trained Momentum ResNets are promising for fine-tuning models.

Contents

4.1 Introduction

Problem setup. As a particular instance of deep learning (LeCun et al., 2015; Goodfellow et al., 2016b), residual neural networks (He et al., 2016a, ResNets) have achieved great empirical successes due to extremely deep representations and their extensions keep on outperforming state of the art on real data sets (Kolesnikov et al., 2019; Touvron et al., 2019). Most of deep learning tasks involve graphics processing units (GPUs), where memory is a practical bottleneck in several situations (Wang et al., 2018; Peng et al., 2017; Zhu et al., 2017). Indeed, backpropagation, used for optimizing deep architectures, requires to store values (activations) at each layer during the evaluation of the network (forward pass). Thus, the depth of deep architectures is constrained by the amount of available memory. The main goal of this chapter is to explore the properties of a new model, Momentum ResNets, that circumvent these memory issues by being invertible: the activations at layer n is recovered exactly from activations at layer $n+1$. This network relies on a modification of the ResNet's forward rule which makes it exactly invertible in practice. Instead of considering the feedforward relation for a ResNet (residual building block)

$$x_{n+1} = x_n + f(x_n, \theta_n), \tag{4.1}$$

we define its momentum counterpart, which iterates

$$\begin{cases} v_{n+1} = \gamma v_n + (1-\gamma) f(x_n, \theta_n) \\ x_{n+1} = x_n + v_{n+1}, \end{cases} \tag{4.2}$$

where f is a parameterized function, v is a velocity term and $\gamma \in [0, 1]$ is a momentum term. This radically changes the dynamics of the network, as shown in the following figure.

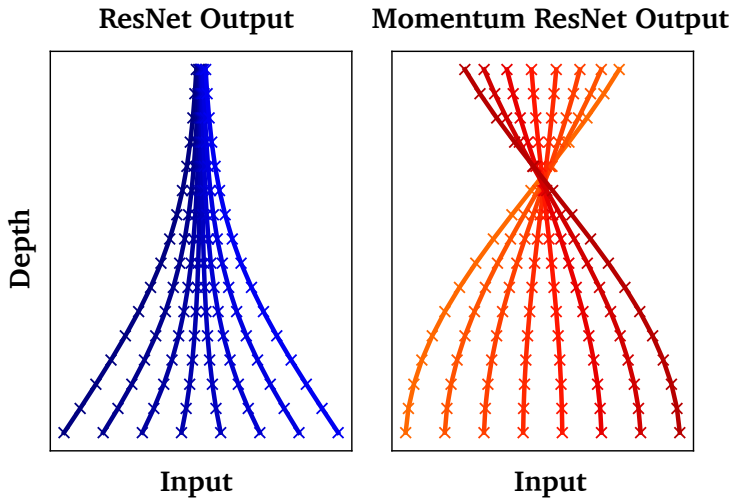

Figure 4.1: **Comparison of the dynamics** of a ResNet (left) and a Momentum ResNet with $\gamma = 0.9$ (right) with **tied weights** between layers, $\theta_n = \theta$ for all n. The evolution of the activations at each layer is shown (depth 15). Models try to learn the mapping $x \mapsto -x^3$ in $\mathbb{R}$. The ResNet fails (the iterations approximate the solution of a first-order ODE, for which trajectories don't cross, cf. Picard-Lindelof theorem) while the Momentum ResNet leverages the changes in velocity to model more complex dynamics.

In contrast with existing reversible models, Momentum ResNets can be integrated seamlessly in any deep architecture which uses residual blocks as building blocks (cf. in Section 4.3).

Contributions. We introduce momentum residual neural networks (Momentum ResNets), a new deep model that relies on a simple modification of the ResNet forward rule and which, without any constraint on its architecture, is perfectly invertible. We show that the memory requirement of Momentum ResNets is arbitrarily reduced by changing the momentum term γ (Section 4.3.2), and show that they can be used as a drop-in replacement for traditional ResNets.

On the theoretical side, we show that Momentum ResNets are easily used in the learning to optimize setting, where other reversible models fail to converge (Section 4.3.3). We also investigate the approximation capabilities of Momentum ResNets, seen in the continuous limit as second-order ODEs (Section 4.4). We first show in Proposition 4.2 that Momentum ResNets can represent a strictly larger class of functions than first-order neural ODEs. Then, we give more detailed insights by studying the linear case, where we formally prove in Theorem 4.4 that Momentum ResNets with linear residual functions have universal approximation capabilities, and precisely quantify how the set of representable mappings for such models grows as the momentum term γ increases. This theoretical result is a first step towards a theoretical analysis of representation capabilities of Momentum ResNets.

Our last contribution is the experimental validation of Momentum ResNets on various learning tasks. We first show that Momentum ResNets separate point clouds that ResNets fail to separate (Section 4.5.1). We also show on image datasets (CIFAR-10, CIFAR-100, ImageNet) that Momentum ResNets have similar accuracy as ResNets, with a smaller memory cost (Section 4.5.2). We also show that parameters of a pre-trained model are easily transferred to a Momentum ResNet which achieves comparable accuracy in only few epochs of training. We argue that this way to obtain pre-trained Momentum ResNets is of major importance for fine-tuning a network

on new data for which memory storage is a bottleneck. We provide a Pytorch package with a method `transform(model)` that takes a torchvision ResNet model and returns its Momentum counterpart that achieves similar accuracy with very little refit. We also experimentally validate our theoretical findings in the learning to optimize setting, by confirming that Momentum ResNets perform better than RevNets (Gomez et al., 2017). Our code is available at `https://github.com/michaelsdr/momentumnet`.

4.2 Background and previous works.

Backpropagation. *Backpropagation* is the method of choice to compute the gradient of a scalar-valued function. It operates using the chain rule with a backward traversal of the computational graph (Bauer, 1974). It is also known as reverse-mode automatic differentiation (Baydin et al., 2018; Rumelhart et al., 1986; Verma, 2000; Griewank and Walther, 2008a). The computational cost is similar to the one of evaluating the function itself. The only way to back-propagate gradients through a neural architecture without further assumptions is to store all the intermediate activations during the forward pass. This is the method used in common deep learning libraries such as Pytorch (Paszke et al., 2017), Tensorflow (Abadi et al., 2016) and JAX (Jacobsen et al., 2018). A common way to reduce this memory storage is to use checkpointing: activations are only stored at some steps and the others are recomputed between these check-points as they become needed in the backward pass (e.g., Martens and Sutskever (2012)).

Reversible architectures. However, models that allow backpropagation without storing any activations have recently been developed. They are based on two kinds of approaches. The first is *discrete* and relies on finding ways to easily invert the rule linking activation n to activation $n+1$ (Gomez et al., 2017; Chang et al., 2018; Haber and Ruthotto, 2017b; Jacobsen et al., 2018; Behrmann et al., 2019). In this way, it is possible to recompute the activations *on the fly* during the backward pass: activations do not have to be stored. However, these methods either rely on restricted architectures where there is no straightforward way to transfer a well performing non-reversible model into a reversible one, or do not offer a fast inversion scheme when recomputing activations backward. In contrast, our proposal can be applied to any existing ResNet and is easily inverted. The second kind of approach is *continuous* and relies on ordinary differential equations (ODEs), where ResNets are interpreted as continuous dynamical systems (Weinan, 2017b; Chen et al., 2018; Teh et al., 2019; Sun et al., 2018; Weinan et al., 2019; Lu et al., 2018; Ruthotto and Haber, 2019). This allows one to import theoretical and numerical advances from ODEs to deep learning. These models are often called neural ODEs (Chen et al., 2018) and can be trained by using an adjoint sensitivity method (Pontryagin, 1987), solving ODEs backward in time. This strategy avoids performing reverse-mode automatic differentiation through the operations of the ODE solver and leads to a $O(1)$ memory footprint. However, defining the neural ODE counterpart of an existing residual architecture is not straightforward: optimizing ODE blocks is an infinite dimensional problem requiring a non-trivial time discretization, and the performances of neural ODEs depend on the numerical integrator for the ODE (Gusak et al., 2020). In addition, ODEs cannot always be numerically reversed, because of stability issues: numerical errors can occur and accumulate when a system is run backwards (Gholami et al., 2019; Teh et al., 2019). Thus, in practice, neural ODEs are seldom used in standard deep learning settings. Nevertheless, recent works (Zhang et al., 2019b; Queiruga et al., 2020) incorporate ODE blocks in neural architectures to achieve comparable accuracies to ResNets on CIFAR.

Representation capabilities. Studying the representation capabilities of such models is also important, as it gives insights regarding their performance on real world data. It is well-known that a single residual block has universal approximation capabilities (Cybenko, 1989), meaning that on a compact set any continuous function can be uniformly approximated with a one-layer feedforward fully-connected neural network. However, neural ODEs have limited representation capabilities. Teh et al. (2019) propose to lift points in higher dimensions by concatenating vector fields of data with zeros in an extra-dimensional space, and show that the resulting augmented neural ODEs (ANODEs) achieve lower loss and better generalization on image classification and toy experiments. Li et al. (2019) show that, if the output of the ODE-Net is composed with elements of a terminal family, then universal approximation capabilities are obtained for the convergence in L^p norm for $p < +\infty$, which is insufficient (Teshima et al., 2020). In this work, we consider the representation capabilities in L^∞ norm of the ODEs derived from the forward iterations of a ResNet. Furthermore, Zhang et al. (2020b) proved that doubling the dimension of the ODE leads to universal approximators, although this result has no application in deep learning to our knowledge. In this work, we show that in the continuous limit, our architecture has better representation capabilities than Neural ODEs. We also prove its universality in the linear case.

Momentum in deep networks. Some recent works (He et al., 2020; Chun et al., 2020; Nguyen et al., 2020; Li et al., 2018) have explored momentum in deep architectures. However, these methods differ from ours in their architecture and purpose. Chun et al. (2020) introduce a momentum to solve an optimization problem for which the iterations do not correspond to a ResNet. Nguyen et al. (2020) (resp. He et al. (2020)) add momentum in the case of RNNs (different from ResNets) where the weights are tied to alleviate the vanishing gradient issue (resp. link the key and query encoder layers). Li et al. (2018) consider a particular case where the linear layer is tied and is a symmetric definite matrix. In particular, none of the mentioned architectures are invertible, which is one of the main assets of our method.

Second-order models We show that adding a momentum term corresponds to an Euler integration scheme for integrating a second-order ODE. Some recently proposed architectures (Norcliffe et al., 2020; Rusch and Mishra, 2021; Lu et al., 2018; Massaroli et al., 2020) are also motivated by second-order differential equations. Norcliffe et al. (2020) introduce second-order dynamics to model second-order dynamical systems, whereas our model corresponds to a discrete set of equations in the continuous limit. Also, in our method, the neural network only acts on x, so that although momentum increases the dimension to $2d$, the computational burden of a forward pass is the same as a ResNet of dimension d. Rusch and Mishra (2021) propose second-order RNNs, whereas our method deals with ResNets. Finally, the formulation of LM-ResNet in Lu et al. (2018) differs from our forward pass ($x_{n+1} = x_n + \gamma v_n + (1-\gamma) f(x_n, \theta_n)$), even though they both lead to second-order ODEs. Importantly, none of these second-order formulations are invertible.

Notations For $d \in \mathbb{N}^*$, we denote by $\mathbb{R}^{d\times d}$, $\mathrm{GL}_d(\mathbb{R})$ and $\mathrm{D}_d^{\mathbb{C}}(\mathbb{R})$ the set of real matrices, of invertible matrices, and of **real** matrices that are diagonalizable in $\mathbb{C}$.

4.3 Momentum Residual Neural Networks

We now introduce Momentum ResNet, a simple transformation of **any** ResNet into a model with a small memory requirement, and that can be seen in the continuous limit as a second-order ODE.

4.3.1 Momentum ResNets

Adding a momentum term in the ResNet equations. For **any** ResNet which iterates (4.1), we define its Momentum counterpart, which iterates (4.2), where $(v_n)_n$ is the velocity initialized with some value v_0 in $\mathbb{R}^d$, and $\gamma \in [0, 1]$ is the so-called momentum term. This approach generalizes gradient descent algorithm with momentum (Ruder, 2016), for which f is the gradient of a function to minimize.

Initial speed and momentum term. In this chapter, we consider initial speeds v_0 that depend on x_0 through a simple relation. The simplest options are to set $v_0 = 0$ or $v_0 = f(x_0, \theta_0)$. We prove in Section 4.4 that this dependency between v_0 and x_0 has an influence on the set of mappings that Momentum ResNets can represent. The parameter γ controls how much a Momentum ResNet diverges from a ResNet, and also the amount of memory saving. The closer γ is to 0, the closer Momentum ResNets are to ResNets, but the less memory is saved. In our experiments, we use $\gamma = 0.9$, which we find to work well in various applications.

Invertibility. Procedure (4.2) is inverted through

$$\begin{cases} x_n = x_{n+1} - v_{n+1}, \\ v_n = \frac{1}{\gamma}\left(v_{n+1} - (1-\gamma) f(x_n, \theta_n)\right), \end{cases} \tag{4.3}$$

so that activations can be reconstructed on the fly during the backward pass in a Momentum ResNet. In practice, in order to exactly reverse the dynamics, the information lost by the finite-precision multiplication by γ in (4.2) has to be efficiently stored. We used the algorithm from Maclaurin et al. (2015) to perform this reversible multiplication. It consists in maintaining an information buffer, that is, an integer that stores the bits that are lost at each iteration, so that multiplication becomes reversible. Note that there is always a small loss of floating point precision due to the addition of the learnable mapping f. In practice, we never found it to be a problem: this loss in precision can be neglected compared to the one due to the multiplication by γ.

Table 4.1: **Comparison of reversible residual architectures**

	Neur.ODE	*i-ResNet*	*i-RevNet*	*RevNet*	***Mom.Net***
Closed-form inversion	✓	✗	✓	✓	✓
Same parameters	✗	✓	✗	✗	✓
Unconstrained training	✓	✗	✓	✓	✓

Drop-in replacement. Our approach makes it possible to turn any existing ResNet into a reversible one. In other words, a ResNet can be transformed into its Momentum counterpart without changing the structure of each layer. For instance, consider a ResNet-152 (He et al., 2016a). It is made of 4 layers (of depth 3, 8, 36 and 3) and can easily be turned into its Momentum ResNet counterpart by changing the forward equations (4.1) into (4.2) in the 4 layers. No further change is needed and Momentum ResNets take the exact same parameters as inputs: they are a *drop-in replacement*. This is not the case of other reversible models. Neural ODEs (Chen et al., 2018) take continuous parameters as inputs. i-ResNets (Behrmann et al., 2019) cannot be trained by plain SGD since the spectral norm of the weights requires constrained optimization.

i-RevNets (Jacobsen et al., 2018) and RevNets (Gomez et al., 2017) require to train two networks with their own parameters for each residual block, split the inputs across convolutional channels, and are half as deep as ResNets: they do not take the same parameters as inputs. Table 4.1 summarizes the properties of reversible residual architectures. We discuss in further details the differences between RevNets and Momentum ResNets in sections 4.3.3 and 4.5.3.

4.3.2 Memory cost

Instead of storing the full data at each layer, we only need to store the bits lost at each multiplication by γ (cf. "intertibility"). For an architecture of depth k, this corresponds to storing $\log_2((\frac{1}{\gamma})^k)$ values for each sample ($\frac{k(1-\gamma)}{\ln(2)}$ if γ is close to 1). To illustrate, we consider two situations where storing the activations is by far the main memory bottleneck. First, consider a toy feedforward architecture where $f(x, \theta) = W_2^T \sigma(W_1 x + b)$, with $x \in \mathbb{R}^d$ and $\theta = (W_1, W_2, b)$, where $W_1, W_2 \in \mathbb{R}^{p \times d}$ and $b \in \mathbb{R}^p$, with a depth $k \in \mathbb{N}$. We suppose that the weights are the same at each layer. The training set is composed of n vectors $x^1, ..., x^n \in \mathbb{R}^d$. For **ResNets**, we need to store the weights of the network and the values of all activations for the training set at each layer of the network. In total, the memory needed is $O(k \times d \times n_{batch})$ per iteration. In the case of **Momentum ResNets**, if γ is close to 1 we get a memory requirement of $O((1-\gamma) \times k \times d \times n_{batch})$. This proves that the memory dependency in the depth k is arbitrarily reduced by changing the momentum γ. The memory savings are confirmed in practice, as shown in Figure 4.2.

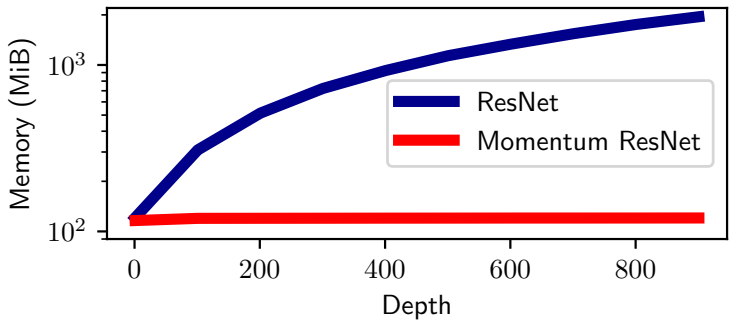

Figure 4.2: **Comparison of memory needed** (calculated using a profiler) for computing gradients of the loss, with ResNets (activations are stored) and Momentum ResNets (activations are not stored). We set $n_{batch} = 500$, $d = 500$ and $\gamma = 1 - \frac{1}{50k}$ at each depth. Momentum ResNets give a nearly constant memory footprint.

As another example, consider a ResNet-152 (He et al., 2016a) which can be used for ImageNet classification (Deng et al., 2009b). Its layer named "`conv4_x`" has a depth of 36: it has 40 M parameters, whereas storing the activations would require storing 50 times more parameters. Since storing the activations is here the main obstruction, the memory requirement for this layer can be arbitrarily reduced by taking γ close to 1.

4.3.3 The role of momentum

When γ is set to 0 in (4.2), we recover a ResNet. Therefore, Momentum ResNets are a generalization of ResNets. When $\gamma \to 1$, one can scale $f \to \frac{1}{1-\gamma} f$ to get in (4.2) a symplectic scheme (Hairer et al., 2006) that recovers a special case of other popular invertible neural network: RevNets (Gomez et al., 2017) and Hamiltonian Networks (Chang et al., 2018). A RevNet

iterates

$$v_{n+1} = v_n + \varphi(x_n, \theta_n), \quad x_{n+1} = x_n + \psi(v_{n+1}, \theta_n'), \tag{4.4}$$

where φ and ψ are two learnable functions.

The usefulness of such architecture depends on the task. RevNets have encountered success for classification and regression. However, we argue that RevNets cannot work in some settings. For instance, under mild assumptions, the RevNet iterations do not have attractive fixed points when the parameters are the same at each layer: $\theta_n = \theta$, $\theta_n' = \theta'$. We rewrite (4.4) as $(v_{n+1}, x_{n+1}) = \Psi(v_n, x_n)$ with $\Psi(v, x) = (v + \varphi(x, \theta), x + \psi(v + \varphi(x, \theta), \theta'))$.

Proposition 4.1 (Instability of fixed points)**.** *Let (v^*, x^*) a fixed point of the RevNet iteration (4.4). Assume that φ (resp. ψ) is differentiable at x^* (resp. v^*), with Jacobian matrix A (resp. B) $\in \mathbb{R}^{d\times d}$. The Jacobian of Ψ at (v^*, x^*) is $J(A, B) = \left(\begin{smallmatrix} \mathrm{Id}_d & A \\ B & \mathrm{Id}_d + BA \end{smallmatrix}\right)$. If A and B are invertible, then there exists $\lambda \in \mathrm{Sp}\,(J(A, B))$ such that $|\lambda| \geq 1$ and $\lambda \neq 1$.*

This shows that (v^*, x^*) cannot be a stable fixed point. As a consequence, in practice, a RevNet cannot have converging iterations: according to (4.4), if x_n converges then v_n must also converge, and their limit must be a fixed point. The previous proposition shows that it is impossible.

This result suggests that RevNets should perform poorly in problems where one expects the iterations of the network to converge. For instance, as shown in the experiments in Section 4.5.3, this happens when we use reverible dynamics in order to *learn to optimize* (Maclaurin et al., 2015). In contrast, the proposed method can converge to a fixed point as long as the momentum term γ is strictly less than 1.

Remark. Proposition 4.1 has a continuous counterpart. Indeed, in the continuous limit, (4.4) writes $\dot{v} = \varphi(x, \theta), \quad \dot{x} = \psi(v, \theta')$. The corresponding Jacobian in (v^*, x^*) is $\left(\begin{smallmatrix} 0 & A \\ B & 0 \end{smallmatrix}\right)$. The eigenvalues of this matrix are the square roots of those of AB: they cannot all have a real part < 0 (same stability issue in the continuous case).

4.3.4 Momentum ResNets as continuous models

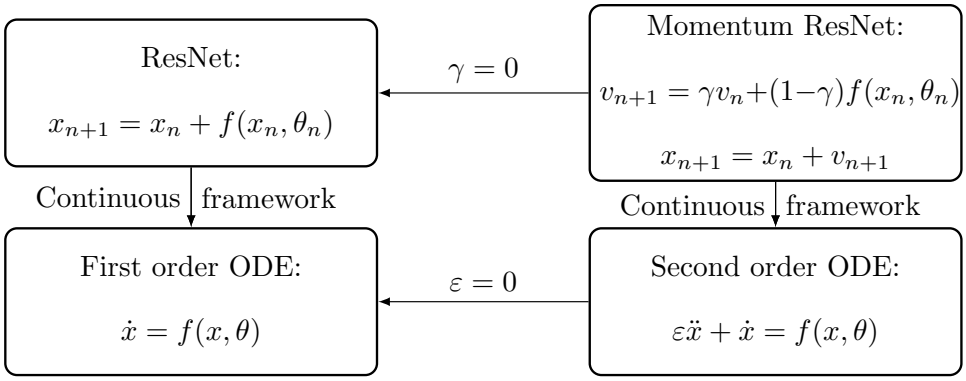

Figure 4.3: **Overview of the four different paradigms.**

Neural ODEs: ResNets as first-order ODEs. The ResNets equation (4.1) with initial condition x_0 (the input of the ResNet) can be seen as a discretized Euler scheme of the ODE $\dot{x} = f(x, \theta)$ with $x(0) = x_0$. Denoting T a time horizon, the neural ODE maps the input $x(0)$ to the output $x(T)$, and, as in Chen et al. (2018), is trained by minimizing a loss $L(x(T), \theta)$.

Momentum ResNets as second-order ODEs. Let $\varepsilon = \frac{1}{1-\gamma}$. We can then rewrite (4.2) as

$$v_{n+1} = v_n + \frac{f(x_n, \theta_n) - v_n}{\varepsilon}, \quad x_{n+1} = x_n + v_{n+1},$$

which corresponds to a Verlet integration scheme (Hairer et al., 2006) with step size 1 of the differential equation $\varepsilon\ddot{x} + \dot{x} = f(x, \theta)$. Thus, in the same way that ResNets can be seen as discretization of first-order ODEs, Momentum ResNets can be seen as discretization of second-order ones. Figure 4.3 sums up these ideas.

4.4 Representation capabilities

We now turn to the analysis of the representation capabilities of Momentum ResNets in the continuous setting. In particular, we precisely characterize the set of mappings representable by Momentum ResNets with linear residual functions.

4.4.1 Representation capabilities of first-order ODEs

We consider the first-order model

$$\dot{x} = f(x, \theta) \quad \text{with} \quad x(0) = x_0. \tag{4.5}$$

We denote by $\varphi_t(x_0)$ the solution at time t starting at initial condition $x(0) = x_0$. It is called the *flow* of the ODE. For all $t \in [0, T]$, where T is a time horizon, φ_t is a homeomorphism: it is continuous, bijective with continuous inverse.

First-order ODEs are not universal approximators. ODEs such as (4.5) are not universal approximators. Indeed, the function mapping an initial condition to the flow at a certain time horizon T cannot represent every mapping $x_0 \mapsto h(x_0)$. For instance when $d = 1$, the mapping $x \to -x$ cannot be approximated by a first-order ODE, since 1 should be mapped to -1 and 0 to 0, which is impossible without intersecting trajectories (Teh et al., 2019). In fact, the homeomorphisms represented by (4.5) are orientation-preserving: if $K \subset \mathbb{R}^d$ is a compact set and $h : K \to \mathbb{R}^d$ is a homeomorphism represented by (4.5), then h is in the connected component of the identity function on K for the topology of the uniform convergence (see details in Appendix 4.B.5).

4.4.2 Representation capabilities of second-order ODEs

We consider the second-order model for which we recall that Momentum ResNets are a discretization:

$$\varepsilon\ddot{x} + \dot{x} = f(x, \theta) \quad \text{with} \quad (x(0), \dot{x}(0)) = (x_0, v_0). \tag{4.6}$$

Representation capabilities of a model (4.6) on the x space. We recall that we consider initial speeds v_0 that can depend on the input $x_0 \in \mathbb{R}^d$ (for instance $v_0 = 0$ or $v_0 = f(x_0, \theta_0)$). We therefore assume $\varphi_t : \mathbb{R}^d \mapsto \mathbb{R}^d$ such that $\varphi_t(x_0)$ is solution of (4.6). We emphasize that φ_t is not always a homeomorphism. For instance, $\varphi_t(x_0) = x_0 \exp(-t/2)\cos(t/2)$ solves $\ddot{x} + \dot{x} = -\frac{1}{2}x(t)$ with $(x(0), \dot{x}(0)) = (x_0, -\frac{x_0}{2})$. All the trajectories intersect at time π. It means that Momentum ResNets can learn mappings that are not homeomorphisms, which suggests that increasing ε should lead to better representation capabilities. The first natural question is thus whether, given $h : \mathbb{R}^d \to \mathbb{R}^d$, there exists some f such that φ_t associated to (4.6) satisfies $\forall x \in \mathbb{R}^d, \varphi_1(x) = h(x)$. In the case where v_0 is an arbitrary function of x_0, the answer is trivial since (4.6) can represent

any mapping, as proved in Appendix 4.B.2. This setting does not correspond to the common use case of ResNets, which take advantage of their depth, so it is important to impose stronger constraints on the dependency between v_0 and x_0. For instance, the next proposition shows that even if one imposes $v_0 = f(x_0, \theta_0)$, a second-order model is at least as general as a first-order one.

Proposition 4.2 (Momentum ResNets are at least as general)**.** *There exists a function $\hat{f}$ such that for all x solution of (4.5), x is also solution of the second-order model $\varepsilon\ddot{x} + \dot{x} = \hat{f}(x, \theta)$ with $(x(0), \dot{x}(0)) = (x_0, f(x_0, \theta_0))$.*

Furthermore, even with the restrictive initial condition $v_0 = 0$, $x \mapsto \lambda x$ for $\lambda > -1$ can always be represented by a second-order model (4.6) (see details in Appendix 4.B.4). This supports the claim that the set of representable mappings increases with ε.

4.4.3 Universality of Momentum ResNets with linear residual functions

As a first step towards a theoretical analysis of the universal representation capabilities of Momentum ResNets, we now investigate the linear residual function case. Consider the second-order linear ODE

$$\varepsilon\ddot{x} + \dot{x} = \theta x \quad \text{with} \quad (x(0), \dot{x}(0)) = (x_0, 0), \tag{4.7}$$

with $\theta \in \mathbb{R}^{d\times d}$. We assume without loss of generality that the time horizon is $T = 1$. We have the following result.

Proposition 4.3 (Solution of (4.7))**.** *At time 1, (4.7) defines the linear mapping $x_0 \mapsto \varphi_1(x_0) = \Psi_\varepsilon(\theta)x_0$ where*

$$\Psi_\varepsilon(\theta) = e^{-\frac{1}{2\varepsilon}} \sum_{n=0}^{+\infty} \left(\frac{1}{(2n)!} + \frac{1}{2\varepsilon(2n+1)!} \right) \left(\frac{\theta}{\varepsilon} + \frac{\mathrm{Id}_d}{4\varepsilon^2} \right)^n.$$

Characterizing the set of mappings representable by (4.7) is thus equivalent to precisely analyzing the range $\Psi_\varepsilon(\mathbb{R}^{d\times d})$.

Representable mappings of a first-order linear model. When $\varepsilon \to 0$, $\Psi_\varepsilon(\theta) \to \Psi_0(\theta) = \exp\theta$. The range of the matrix exponential is indeed the set of representable mappings of a first order linear model

$$\dot{x} = \theta x \quad \text{with} \quad x(0) = x_0 \tag{4.8}$$

and this range is known (Andrica and Rohan, 2010) to be $\Psi_0(\mathbb{R}^{d\times d}) = \exp(\mathbb{R}^{d\times d}) = \{M^2 \mid M \in \mathrm{GL}_d(\mathbb{R})\}$. This means that one can only learn mappings that are the square of invertible mappings with a first-order linear model (4.8). To ease the exposition and exemplify the impact of increasing $\varepsilon > 0$, we now consider the case of matrices with real coefficients that are diagonalizable in $\mathbb{C}$, $\mathrm{D}_d^{\mathbb{C}}(\mathbb{R})$. Note that the general setting of arbitrary matrices is exposed in Appendix 4.A.3 using Jordan decomposition. Note also that $\mathrm{D}_d^{\mathbb{C}}(\mathbb{R})$ is dense in $\mathbb{R}^{d\times d}$ (Hartfiel, 1995). Using Theorem 1 from Culver (1966), we have that if $D \in \mathrm{D}_d^{\mathbb{C}}(\mathbb{R})$, then D is represented by a first-order model (4.8) **if and only if** D is non-singular and for all eigenvalues $\lambda \in \mathrm{Sp}(D)$ with $\lambda < 0$, λ is of even multiplicity order. This is restrictive because it forces negative eigenvalues to be in pairs. We now generalize this result and show that increasing $\varepsilon > 0$ leads to less restrictive conditions.

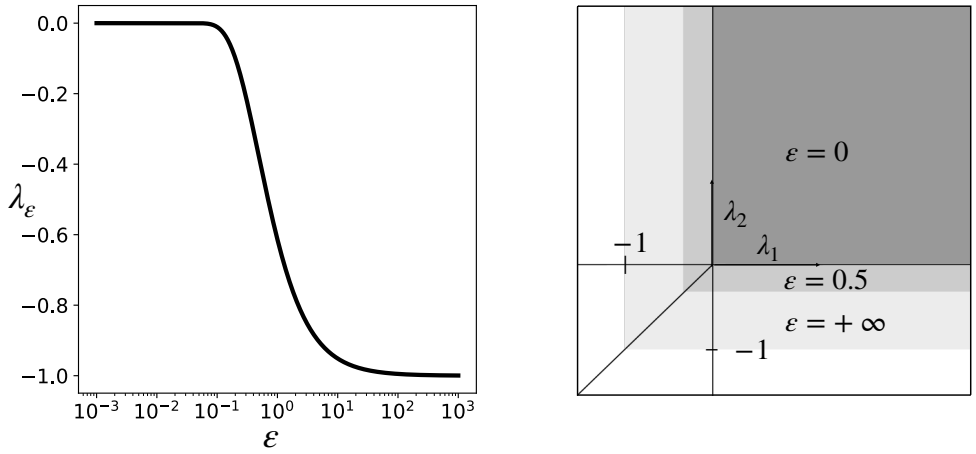

Figure 4.4: Left: **Evolution of λ_ε defined in Theorem 4.4**. λ_ε is non increasing, stays close to 0 when $\varepsilon \ll 1$ and close to -1 when $\varepsilon \geq 2$. Right: **Evolution of the real eigenvalues** λ_1 and λ_2 of representable matrices in $\mathrm{D}_d^{\mathbb{C}}(\mathbb{R})$ by (4.7) when $d = 2$ for different values of ε. The grey colored areas correspond to the different representable eigenvalues. When $\varepsilon = 0$, $\lambda_1 = \lambda_2$ or $\lambda_1 > 0$ and $\lambda_2 > 0$. When $\varepsilon > 0$, single negative eigenvalues are acceptable.

Representable mappings by a second-order linear model. Again, by density and for simplicity, we focus on matrices in $\mathrm{D}_d^{\mathbb{C}}(\mathbb{R})$, and we state and prove the general case in Appendix 4.A.3, making use of Jordan blocks decomposition of matrix functions (Gantmacher, 1959) and localization of zeros of entire functions (Runckel, 1969). The range of Ψ_ε over the reals has for form $\Psi_\varepsilon(\mathbb{R}) = [\lambda_\varepsilon, +\infty[$. It plays a pivotal role to control the set of representable mappings, as stated in the theorem bellow. Its minimum value can be computed conveniently since it satisfies $\lambda_\varepsilon = \min_{\alpha \in \mathbb{R}} G_\varepsilon(\alpha)$ where $G_\varepsilon(\alpha) \triangleq \exp\left(-\frac{1}{2\varepsilon}\right)(\cos(\alpha) + \frac{1}{2\varepsilon\alpha}\sin(\alpha))$.

Theorem 4.4 (Representable mappings with linear residual functions)**.** *Let $D \in \mathrm{D}_d^{\mathbb{C}}(\mathbb{R})$. Then D is represented by a second-order model (4.7) **if and only if** $\forall \lambda \in \mathrm{Sp}(D)$ such that $\lambda < \lambda_\varepsilon$, λ is of even multiplicity order.*

Theorem 4.4 is illustrated Figure 4.4. A consequence of this result is that the set of representable linear mappings is **strictly increasing** with ε. Another consequence is that one can learn **any** mapping up to scale using the ODE (4.7): if $D \in \mathrm{D}_d^{\mathbb{C}}(\mathbb{R})$, there exists $\alpha_\varepsilon > 0$ such that for all $\lambda \in \mathrm{Sp}(\alpha_\varepsilon D)$, one has $\lambda > \lambda_\varepsilon$. Theorem 4.4 shows that $\alpha_\varepsilon D$ is represented by a second-order model (4.7).

4.5 Experiments

We now demonstrate the applicability of Momentum ResNets through experiments. We used Pytorch and Nvidia Tesla V100 GPUs.

4.5.1 Point clouds separation

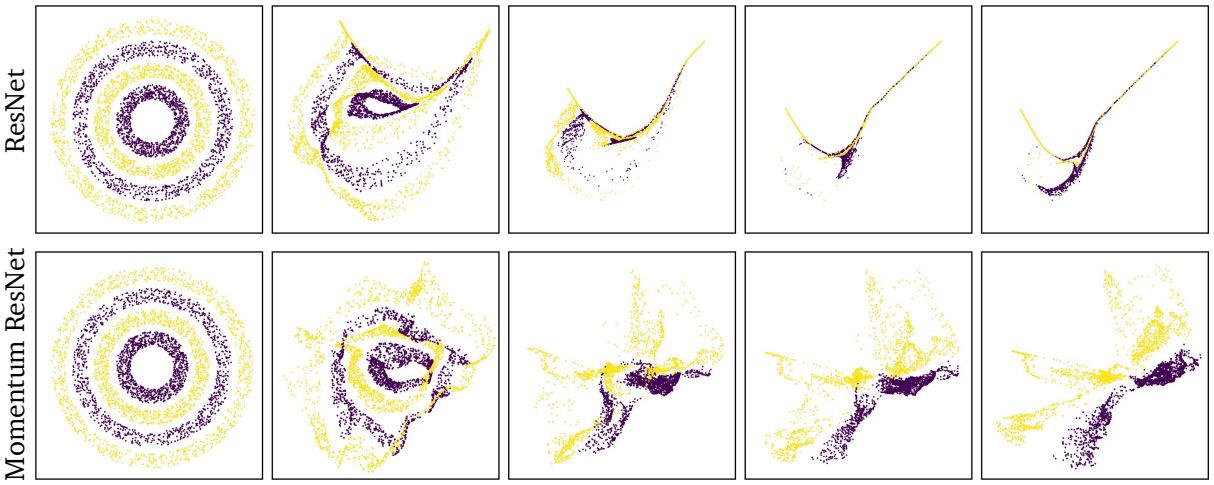

Figure 4.5: **Separation of four nested rings** using a ResNet (upper row) and a Momentum ResNet (lower row). From left to right, each figure represents the point clouds transformed at layer $3k$. The ResNet fails whereas the Momentum ResNet succeeds.

We experimentally validate the representation capabilities of Momentum ResNets on a challenging synthetic classification task. As already noted (Teh et al., 2019), neural ODEs ultimately fail to break apart nested rings. We experimentally demonstrate the advantage of Momentum ResNets by separating 4 nested rings (2 classes). We used the same structure for both models: $f(x, \theta) = W_2^T \tanh(W_1 x + b)$ with W_1, $W_2 \in \mathbb{R}^{16\times 2}$, $b \in \mathbb{R}^{16}$, and a depth 15. Evolution of the points as depth increases is shown in Figure 4.5. The fact that the trajectories corresponding to the ResNet panel don't cross is because, with this depth, the iterations approximate the solution of a first order ODE, for which trajectories cannot cross, due to the Picard-Lindelof theorem.

4.5.2 Image experiments

We also compare the accuracy of ResNets and Momentum ResNets on real data sets: CIFAR-10, CIFAR-100 (Krizhevsky et al., 2010) and ImageNet (Deng et al., 2009b). We used existing ResNets architectures. We recall that Momentum ResNets can be used as a drop-in replacement and that it is sufficient to replace every residual building block with a momentum residual forward iteration. We set $\gamma = 0.9$ in the experiments. More details about the experimental setup are given in Appendix 4.D.

Table 4.2: **Test accuracy for CIFAR** over 10 runs for each model

Model	**CIFAR-10**	**CIFAR-100**
Momentum ResNet, $v_0 = 0$	95.1 ± 0.13	76.39 ± 0.18
Momentum ResNet, $v_0 = f(x_0)$	95.18 ± 0.06	76.38 ± 0.42
ResNet	95.15 ± 0.12	76.86 ± 0.25

Results on CIFAR-10 and CIFAR-100. For these data sets, we used a ResNet-101 (He et al., 2016a) and a Momentum ResNet-101 and compared the evolution of the test error and

test loss. Two kinds of Momentum ResNets were used: one with an initial speed $v_0 = 0$ and the other one where the initial speed v_0 was learned: $v_0 = f(x_0)$. These experiments show that Momentum ResNets perform similarly to ResNets. Results are summarized in Table 4.2.

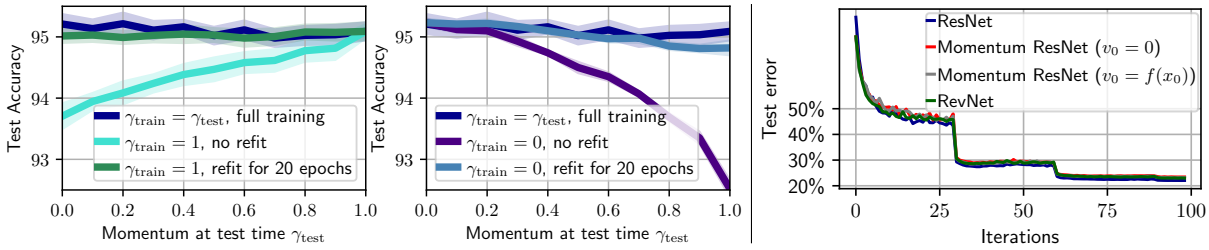

Figure 4.6: Upper row: **Robustness of final accuracy w.r.t** γ when training Momentum ResNets 101 on CIFAR-10. We train the networks with a momentum γ_{train} and evaluate their accuracy with a different momentum γ_{test} at test time. We optionally refit the networks for 20 epochs. We recall that $\gamma_{\text{train}} = 0$ corresponds to a classical ResNet and $\gamma_{\text{train}} = 1$ corresponds to a Momentum ResNet with optimal memory savings. Lower row: **Top-1 classification error on ImageNet (single crop)** for 4 different residual architectures of depth 101 with the same number of parameters. Final test accuracy is 22% for the ResNet-101 and 23% for the 3 other invertible models. In particular, our model achieve the same performance as a RevNet with the same number of parameters.

Effect of the momentum term γ. Theorem 4.4 shows the effect of ε on the representable mappings for linear ODEs. To experimentally validate the impact of γ, we train a Momentum ResNet-101 on CIFAR-10 for different values of the momentum at train time, γ_{train}. We also evaluate Momentum ResNets trained with $\gamma_{\text{train}} = 0$ and $\gamma_{\text{train}} = 1$ with no further training for several values of the momentum at test time, γ_{test}. In this case, the test accuracy never decreases by more than 3%. We also refit for 20 epochs Momentum ResNets trained with $\gamma_{\text{train}} = 0$ and $\gamma_{\text{train}} = 1$. This is sufficient to obtain similar accuracy as models trained from scratch. Results are shown in Figure 4.6 (upper row). This indicates that the choice of γ has a limited impact on accuracy. In addition, learning the parameter γ does not affect the accuracy of the model. Since it also breaks the method described in 4.3.2, we fix γ in all the experiments.

Results on ImageNet. For this data set, we used a ResNet-101, a Momentum ResNet-101, and a RevNet-101. For the latter, we used the procedure from Gomez et al. (2017) and adjusted the depth of each layer for the model to have approximately the same number of parameters as the original ResNet-101. Evolution of test errors are shown in Figure 4.6 (lower row), where comparable performances are achieved.

Memory costs. We compare the memory (using a memory profiler) for performing one epoch as a function of the batch size for two datasets: ImageNet (depth of 152) and CIFAR-10 (depth of 1201). Results are shown in Figure 4.7 and illustrate how Momentum ResNets can benefit from increased batch size, especially for very deep models. We also show in Figure 4.7 the final test accuracy for a full training of Momentum ResNets on CIFAR-10 as a function of the memory used (directly linked to γ (section 4.3.2)).

Ability to perform pre-training and fine-tuning. It has been shown (Tajbakhsh et al., 2016) that in various medical imaging applications the use of a pre-trained model on ImageNet with adapted fine-tuning outperformed a model trained from scratch. In order to easily obtain pre-trained Momentum ResNets for applications where memory could be a bottleneck, we transferred

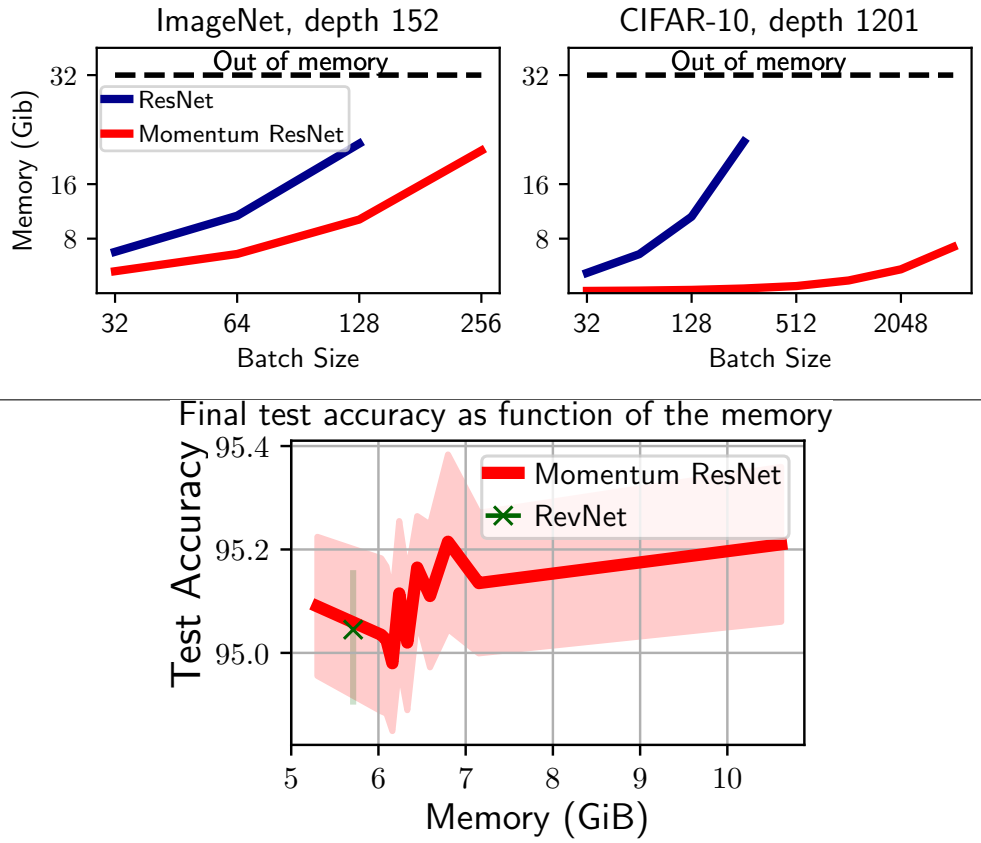

Figure 4.7: Upper row: **Memory used** (using a profiler) for a ResNet and a Momentum ResNet on one training epoch, as a function of the batch size. Lower row: **Final test accuracy** as a function of the memory used (per epoch) for training Momentum ResNets-101 on CIFAR-10.

the learned parameters of a ResNet-152 pre-trained on ImageNet to a Momentum ResNet-152 with $\gamma = 0.9$. In only 1 epoch of additional training we reached a top-1 error of 26.5% and in 5 additional epochs a top-1 error of 23.5%. We then empirically compared the accuracy of these pre-trained models by fine-tuning them on new images: the `hymenoptera`[1] data set.

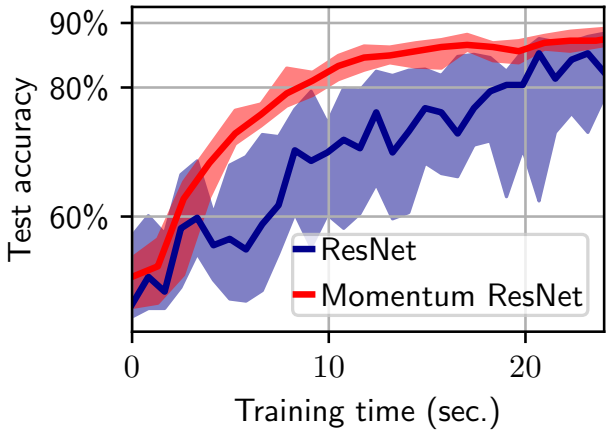

Figure 4.8: **Accuracy** as a function of time on `hymenoptera` when fine-tuning a ResNet-152 and a Momentum ResNet-152 with batch sizes of 2 and 4, respectively, as permitted by memory.

As a proof of concept, suppose we have a GPU with 3 Go of RAM. The images have a resolution of 500×500 pixels so that the maximum batch size that can be taken for fine-tuning the ResNet-152

[1] https://www.kaggle.com/ajayrana/hymenoptera-data

is 2, against 4 for the Momentum ResNet-152. As suggested in Tajbakhsh et al. (2016) ("if the distance between the source and target applications is significant, one may need to fine-tune the early layers as well"), we fine-tune the whole network in this proof of concept experiment. In this setting the Momentum ResNet leads to faster convergence when fine-tuning, as shown in Figure 4.8: Momentum ResNets can be twice as fast as ResNets to train when samples are so big that only few of them can be processed at a time. In contrast, RevNets (Gomez et al., 2017) cannot as easily be used for fine-tuning since, as shown in (4.4), they require to train two distinct networks.

Continuous training. We also compare accuracy when using first-order ODE blocks (Chen et al., 2018) and second-order ones on CIFAR-10. In order to emphasize the influence of the ODE, we considered a neural architecture which down-sampled the input to have a certain number of channels, and then applied 10 successive ODE blocks. Two types of blocks were considered: one corresponded to the first-order ODE (4.5) and the other one to the second-order ODE (4.6). Training was based on the odeint function implemented by Chen et al. (2018). Figure 4.9 shows the final test accuracy for both models as a function of the number of channels used. As a baseline, we also include the final accuracy when there are no ODE blocks. We see that an ODE Net with momentum significantly outperforms an original ODE Net when the number of channels is small. Training took the same time for both models.

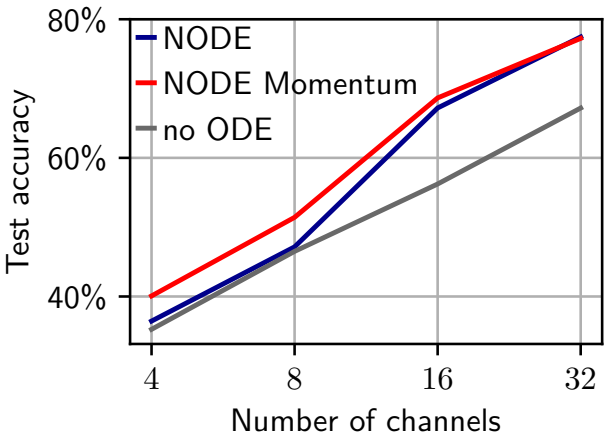

Figure 4.9: **Accuracy after** 120 **iterations on CIFAR-10** with or without momentum, when varying the number of channels.

4.5.3 Learning to optimize

We conclude by illustrating the usefulness of our Momentum ResNets in the *learning to optimize* setting, where one tries to learn to minimize a function. We consider the Learned-ISTA (LISTA) framework (Gregor and LeCun, 2010). Given a matrix $D \in \mathbb{R}^{d \times p}$, and a hyper-parameter $\lambda > 0$, the goal is to perform the sparse coding of a vector $y \in \mathbb{R}^d$, by finding $x \in \mathbb{R}^p$ that minimizes the Lasso cost function $\mathcal{L}_y(x) \triangleq \frac{1}{2}\|y - Dx\|^2 + \lambda\|x\|_1$ (Tibshirani, 1996). In other words, we want to compute a mapping $y \mapsto \operatorname{argmin}_x \mathcal{L}_y(x)$. The ISTA algorithm (Daubechies et al., 2004) solves the problem, starting from $x_0 = 0$, by iterating $x_{n+1} = \mathrm{ST}(x_n - \eta D^\top(Dx_n - y), \eta\lambda)$, with $\eta > 0$ a step-size. Here, ST is the soft-thresholding operator. The idea of Gregor and LeCun (2010) is to view L iterations of ISTA as the output of a neural network with L layers that iterates $x_{n+1} = g(x_n, y, \theta_n) \triangleq \mathrm{ST}(W_n^1 x_n + W_n^2 y, \eta\lambda)$, with parameters $\theta \triangleq (\theta_1, \dots, \theta_L)$ and $\theta_n \triangleq (W_n^1, W_n^2)$. We call $\Phi(y, \theta)$ the network function, which maps y to the output x_L. Importantly, this network can be seen as a residual network, with residual function $f(x, y, \theta) = g(x, y, \theta) - x$. ISTA corresponds to fixed parameters between layers: $W_n^1 = \mathrm{Id}_p - \eta D^\top D$ and $W_n^2 = \eta D^\top$, but these parameters can be learned to yield better performance. We focus on an "unsupervised" learning setting, where

we have some training examples $y^1, \dots, y^Q$, and use them to learn parameters θ that quickly minimize the Lasso function $\mathcal{L}$. In other words, the parameters θ are estimated by minimizing the cost function $\theta \mapsto \sum_{q=1}^{Q} \mathcal{L}_{y_q}(\Phi(y_q, \theta))$. The performance of the network is then measured by computing the testing loss, that is the Lasso loss on some unseen testing examples.

We consider a Momentum ResNet and a RevNet variant of LISTA which use the residual function f. For the RevNet, the activations x_n are first duplicated: the network has twice as many parameters at each layer. The matrix D is generated with i.i.d. Gaussian entries with $p = 32$, $d = 16$, and its columns are then normalized to unit variance. Training and testing samples y are generated as normalized Gaussian i.i.d. entries. More details on the experimental setup are added in Appendix 4.D. The next Figure 4.10 shows the test loss of the different methods, when the depth of the networks varies.

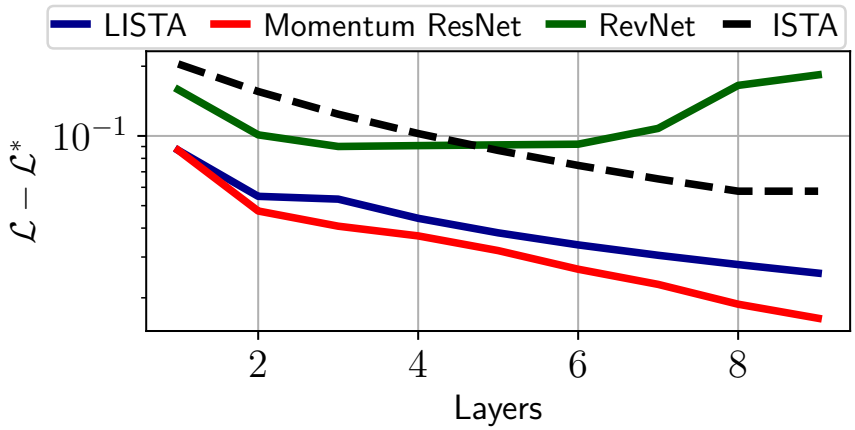

Figure 4.10: **Evolution of the test loss** for different models as a function of depth in the Learned-ISTA (LISTA) framework.

As predicted by Proposition 4.1, the RevNet architecture fails on this task: it cannot have converging iterations, which is exactly what is expected here. In contrast, the Momentum ResNet works well, and even outperforms the LISTA baseline. This is not surprising: it is known that momentum can accelerate convergence of first order optimization methods.

Conclusion

This chapter introduces Momentum ResNets, new invertible residual neural networks operating with a significantly reduced memory footprint compared to ResNets. In sharp contrast with existing invertible architectures, they are made possible by a simple modification of the ResNet forward rule. This simplicity offers both theoretical advantages (better representation capabilities, tractable analysis of linear dynamics) and practical ones (drop-in replacement, speed and memory improvements for model fine-tuning). Momentum ResNets interpolate between ResNets ($\gamma = 0$) and RevNets ($\gamma = 1$), and are a natural second-order extension of neural ODEs. As such, they can capture non-homeomorphic dynamics and converging iterations. As shown in this chapter, the latter is not possible with existing invertible residual networks, although crucial in the learning to optimize setting.

In Section 4.A we give the proofs of all the Propositions and the Theorem. In Section 4.B we give other theoretical results to validate statements made in the chapter. Section 4.C presents the algorithm from Maclaurin et al. (2015). Section 4.D gives details for the experiments in

the chapter. We derive the formula for backpropagation in Momentum ResNets in Section 4.E. Finally, we present additional figures in Section 4.F.

4.A Proofs

Notations

- $C_0^\infty([0,1],\mathbb{R}^d)$ is the set of infinitely differentiable functions from $[0,1]$ to $\mathbb{R}^d$ with value 0 in 0.
- If $f : U \times V \to W$ is a function, we denote by $\partial_u f$, when it exists, the partial derivative of f with respect to $u \in U$.
- For a matrix $A \in \mathbb{R}^{d\times d}$, we denote by $(\lambda - z)^a$ the Jordan block of size $a \in \mathbb{N}$ associated to the eigenvalue $z \in \mathbb{C}$.

4.A.0 Instability of fixed points – Proof of Proposition 4.1

Proof. Since (x^*, v^*) is a fixed point of the RevNet iteration, we have

$$\varphi(x^*) = 0$$

$$\psi(v^*) = 0$$

Then, a first order expansion, writing $x = x^* + \varepsilon$ and $v = v^* + \delta$ gives at order one

$$\Psi(v,x) = (v^* + \delta + A\varepsilon, x^* + \varepsilon + B(\delta + A\varepsilon)) \tag{4.9}$$

We therefore obtain at order one

$$\Psi(v,x) = \Psi(v^*,x^*) + J(A,B)\begin{pmatrix}\delta\\ \varepsilon\end{pmatrix}$$

which shows that $J(A,B)$ is indeed the Jacobian of Ψ at (v^*,x^*). We now turn to a study of the spectrum of $J(A,B)$. We let $\lambda \in \mathbb{C}$ an eigenvalue of $J(A,B)$, and vectors $u \in \mathbb{C}^d$, $w \in \mathbb{C}^d$ such that (u,w) is the corresponding eigenvector, and study the eigenvalue equation

$$J(A,B)\begin{pmatrix}u\\ w\end{pmatrix} = \lambda\begin{pmatrix}u\\ w\end{pmatrix}$$

which gives the two equations

$$u + Aw = \lambda u \tag{4.10}$$

$$w + Bu + BAw = \lambda w \tag{4.11}$$

We start by showing that $\lambda \neq 1$ by contradiction. Indeed, if $\lambda = 1$, then (4.10) gives $Aw = 0$, which implies $w = 0$ since A is invertible. Then, (4.11) gives $Bu = 0$, which also implies $u = 0$. This contradicts the fact that (u,v) is an eigenvector (which is non-zero by definition).

Then, the first equation (4.10) gives $Aw = (\lambda - 1)u$, and multiplying (4.11) by A on the left gives

$$\lambda ABu = (\lambda - 1)^2 u \tag{4.12}$$

We also cannot have $\lambda = 0$, since it would imply $u = 0$. Then, dividing (4.12) by λ shows that $\frac{(\lambda-1)^2}{\lambda}$ is an eigenvalue of AB.

Next, we let $\mu \neq 0$ the eigenvalue of AB such that $\mu = \frac{(\lambda-1)^2}{\lambda}$. The equation can be rewritten as the second order equation

$$\lambda^2 - (2+\mu)\lambda + 1 = 0$$

This equation has two solutions $\lambda_1(\mu)$, $\lambda_2(\mu)$, and since the constant term is 1, we have $\lambda_1(\mu)\lambda_2(\mu) = 1$. Taking modulus, we get $|\lambda_1(\mu)||\lambda_2(\mu)| = 1$, which shows that necessarily, either $|\lambda_1(\mu)| \geq 1$ or $|\lambda_1(\mu)| \geq 1$.

Now, the previous reasoning is only a necessary condition on the eigenvalues, but we can now prove the advertised result by going backwards: we let $\mu \neq 0$ an eigenvalue of AB, and $u \in \mathbb{C}^d$ the associated eigenvector. We consider λ a solution of $\lambda^2 - (2+\mu)\lambda + 1 = 0$ such that $|\lambda| \geq 1$ and $\lambda \neq 1$. Then, we consider $w = (\lambda - 1)A^{-1}u$. We just have to verify that (u, v) is an eigenvector of $J(A, B)$. By construction, (4.10) holds. Next, we have

$$A(w + Bu + BAw) = (\lambda - 1)u + ABu + (\lambda - 1)ABu = (\lambda - 1)u + \lambda ABu$$

Leveraging the fact that u is an eigenvector of AB, we have $\lambda ABu = \lambda\mu u$, and finally:

$$A(w + Bu + BAw) = (\lambda - 1 + \lambda\mu)u = \lambda(\lambda - 1)u = \lambda Aw$$

Which recovers exactly (4.11): λ is indeed an eigenvalue of $J(A, B)$. □

4.A.1 Momentum ResNets are more general than neural ODEs – Proof of Proposition 4.2

Proof. If x satisfies (4.5) we get by derivation that

$$\ddot{x} = \partial_x f(x, \theta) f(x, \theta) + \partial_\theta f(x, \theta)\dot{\theta}$$

Then, if we define $\hat{f}(x, \theta) = \varepsilon[\partial_x f(x, \theta) f(x, \theta) + \partial_\theta f(x, \theta)\dot{\theta}] + f(x, \theta)$, we get that x is also solution of the second-order model $\varepsilon\ddot{x} + \dot{x} = \hat{f}(x, \theta)$ with $(x(0), \dot{x}(0)) = (x_0, f(x_0, \theta_0))$. □

4.A.2 Solution of (4.7) – Proof of Proposition 4.3

(4.7) writes

$$\begin{cases} \dot{x} &= v, \quad x(0) = x_0 \\ \dot{v} &= \frac{\theta x - v}{\varepsilon}, \quad v(0) = 0. \end{cases}$$

For which the solution at time t writes

$$\begin{pmatrix} x(t) \\ v(t) \end{pmatrix} = \exp\begin{pmatrix} 0 & \mathrm{Id}_d t \\ \frac{\theta t}{\varepsilon} & -\frac{\mathrm{Id}_d t}{\varepsilon} \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ 0 \end{pmatrix}.$$

The calculation of this exponential gives

$$x(t) = e^{-\frac{t}{2\varepsilon}} \left(\sum_{n=0}^{+\infty} \frac{1}{(2n)!} (\frac{\theta}{\varepsilon} + \frac{\mathrm{Id}_d}{4\varepsilon^2})^n t^{2n} + \sum_{n=0}^{+\infty} \frac{1}{2\varepsilon(2n+1)!} (\frac{\theta}{\varepsilon} + \frac{\mathrm{Id}_d}{4\varepsilon^2})^n t^{2n+1} \right) x_0.$$

Note that it can be checked directly that this expression satisfies (4.7) by derivations. At time 1 this effectively gives $x(1) = \Psi_\varepsilon(\theta) x_0$.

4.A.3 Representable mappings for a Momentum ResNet with linear residual functions – Proof of Theorem 4.4

In what follows, we denote by f_ε the function of matrices defined by

$$f_\varepsilon(\theta) = \Psi_\varepsilon(\varepsilon\theta - \frac{I}{4\varepsilon}) = e^{-\frac{1}{2\varepsilon}} \sum_{n=0}^{+\infty} \left(\frac{1}{(2n)!} + \frac{1}{2\varepsilon(2n+1)!} \right) \theta^n.$$

Because $\Psi_\varepsilon(\mathbb{R}^{d\times d}) = f_\varepsilon(\mathbb{R}^{d\times d})$, we choose to work on f_ε.

We first need to prove that f_ε is surjective on $\mathbb{C}$.

4.A.3.1 Surjectivity on $\mathbb{C}$ of f_ε

Lemma 4.5 (Surjectivity of f_ε)**.** *For $\varepsilon > 0$, f_ε is surjective on $\mathbb{C}$.*

Proof. Consider

$$\begin{aligned} F_\varepsilon \colon \mathbb{C} &\longrightarrow \mathbb{C} \\ z &\longmapsto e^{-\frac{1}{2\varepsilon}}(\cosh(z) + \frac{1}{2\varepsilon z}\sinh(z)). \end{aligned}$$

For $z \in \mathbb{C}$, we have $f_\varepsilon(z^2) = F_\varepsilon(z)$, and because $z \mapsto z^2$ is surjective on $\mathbb{C}$, it is sufficient to prove that F_ε is surjective on $\mathbb{C}$. Suppose by contradiction that there exists $w \in \mathbb{C}$ such that $\forall z \in \mathbb{C}$, $\exp(\frac{1}{2\varepsilon})F_\varepsilon(z) \neq w$. Then $\exp(\frac{1}{2\varepsilon})F_\varepsilon - w$ is an entire function (Levin, 1996) of order 1 with no zeros. Using Hadamard's factorization theorem (Conway, 2012), this implies that there exists $a, b \in \mathbb{C}$ such that $\forall z \in \mathbb{C}$,

$$\cosh(z) + \frac{\sinh(z)}{2\varepsilon z} - w = \exp(az + b).$$

However, since F_ε is an even function one has that $\forall z \in \mathbb{C}$

$$\exp(az + b) = \exp(-az + b)$$

so that $\forall z \in \mathbb{C}$, $2az \in 2i\pi\mathbb{Z}$. Necessarily, $a = 0$, which is absurd because F_ε is not constant. □

We first prove Theorem 4.4 in the diagonalizable case.

4.A.3.2 Theorem 4.4 in the diagonalizable case

Proof. **Necessity** Suppose that D can be represented by a second-order model (4.7). This means that there exists a real matrix X such that $D = f_\varepsilon(X)$ with X real and

$$f_\varepsilon(X) = e^{-\frac{1}{2\varepsilon}}(\sum_{n=0}^{+\infty} a_n^\varepsilon X^n)$$

with

$$a_n^\varepsilon = \frac{1}{(2n)!} + \frac{1}{2\varepsilon(2n+1)!}.$$

X commutes with D so that there exists $P \in \mathrm{GL}_d(\mathbb{C})$ such that $P^{-1}DP$ is diagonal and $P^{-1}XP$ is triangular. Because $f_\varepsilon(P^{-1}XP) = P^{-1}DP$, we have that $\forall \lambda \in \mathrm{Sp}(D)$, there exists $z \in \mathrm{Sp}(X)$ such that $\lambda = f_\varepsilon(z)$. Because $\lambda < \lambda_\varepsilon$, necessarily, $z \in \mathbb{C} - \mathbb{R}$. In addition, $\lambda = f_\varepsilon(z) = \bar{\lambda} = f_\varepsilon(\bar{z})$. Because X is real, each $z \in \mathrm{Sp}(X)$ must be associated with $\bar{z}$ in $P^{-1}XP$. Thus, λ appears in pairs in $P^{-1}DP$.

Sufficiency Now, suppose that $\forall \lambda \in \mathrm{Sp}(D)$ with $\lambda < \lambda_\varepsilon$, λ is of even multiplicity order. We are going to exhibit a X real such that $D = f_\varepsilon(X)$. Thanks to Lemma 4.5, we have that f_ε is surjective. Let $\lambda \in \mathrm{Sp}(D)$.

- If $\lambda \in \mathbb{R}$ and $\lambda < \lambda_\varepsilon$ or $\lambda \in \mathbb{C} - \mathbb{R}$ then there exists $z \in \mathbb{C} - \mathbb{R}$ by Lemma 4.5 such that $\lambda = f_\varepsilon(z)$.
- If $\lambda \in \mathbb{R}$ and $\lambda \geq \lambda_\varepsilon$, then because f_ε is continuous and goes to infinity when $x \in \mathbb{R}$ goes to infinity, there exists $x \in \mathbb{R}$ such that $\lambda = f_\varepsilon(x)$.

In addition, there exist $(\alpha_1, ..., \alpha_k) \in (\mathbb{C} - \mathbb{R})^k \cup [-\infty, \lambda_\varepsilon[^k$, $(\beta_1, ..., \beta_p) \in [\lambda_\varepsilon, +\infty]^p$ such that

$$D = Q^{-1} \Delta Q,$$

with $Q \in \mathrm{GL}_d(\mathbb{R})$, and

$$\Delta = \begin{pmatrix} P_1^{-1} D_{\alpha_1} P_1 & 0_2 & \cdots & \cdots & \cdots & 0_2 \\ 0_2 & \ddots & \cdots & \cdots & \cdots & 0_2 \\ \vdots & \vdots & P_k^{-1} D_{\alpha_k} P_k & 0_2 & \cdots & 0_2 \\ 0 & \cdots & \cdots & \beta_1 & \cdots & 0 \\ 0 & \cdots & \cdots & 0 & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & \beta_p \end{pmatrix} \in \mathbb{R}^{d \times d}$$

with $P_j \in GL_2(\mathbb{C})$ and $D_{\alpha_j} = \begin{pmatrix} \alpha_j & 0 \\ 0 & \bar{\alpha_j} \end{pmatrix}$.

Let $(z_1, ..., z_k) \in (\mathbb{C} - \mathbb{R})^k$ and $(x_1, ..., x_p) \in \mathbb{R}^p$ be such that $f_\varepsilon(z_j) = \alpha_j$ and $f_\varepsilon(x_j) = \beta_j$. For $1 \leq j \leq k$, one has $P_j^{-1} D_{z_j} P_j \in \mathbb{R}^{2\times 2}$. Indeed, writing $\alpha_j = a_j + ib_j$ with $a_j, b_j \in \mathbb{R}$, the fact that $P_j^{-1} D_{\alpha_j} P_j \in \mathbb{R}^{2\times 2}$ implies that $i \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \in i\mathbb{R}^{2\times 2}$. Writing $z_j = u_j + iv_j$ with $u_j, v_j \in \mathbb{R}$, we get that $P_j^{-1} D_{z_j} P_j \in \mathbb{R}^{2\times 2}$. Then

$$X = Q \begin{pmatrix} P_1^{-1} D_{z_1} P_1 & 0_2 & \cdots & \cdots & \cdots & 0_2 \\ 0_2 & \ddots & \cdots & \cdots & \cdots & 0_2 \\ \vdots & \vdots & P_k^{-1} D_{z_k} P_k & 0_2 & \cdots & 0_2 \\ 0 & \cdots & \cdots & x_1 & \cdots & 0 \\ 0 & \cdots & \cdots & 0 & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & x_p \end{pmatrix} Q^{-1} \in \mathbb{R}^{d \times d}$$

is such that $f_\varepsilon(X) = D$, and D is represented by a second-order model (4.7). □

We now state and demonstrate the general version of Theorem 4.4.

First, we need to demonstrate properties of the complex derivatives of the entire function f_ε.

4.A.3.3 The entire function f_ε has a derivative with no-zeros on $\mathbb{C} - \mathbb{R}$.

Lemma 4.6 (On the zeros of f'_ε)**.** *$\forall z \in \mathbb{C} - \mathbb{R}$ we have $f'_\varepsilon(z) \neq 0$.*

Proof. One has

$$G_\varepsilon(z) = e^{-\frac{1}{2\varepsilon}}(\cos(z) + \frac{1}{2\varepsilon z}\sin(z)) = f_\varepsilon(-z^2)$$

so that $G'_\varepsilon(z) = -2zf'_\varepsilon(-z^2)$ and it is sufficient to prove that the zeros of G'_ε are all real.

We first show that G_ε belongs to the Laguerre-Pólya class (Craven and Csordas, 2002). The Laguerre-Pólya class is the set of entire functions that are the uniform limits on compact sets of $\mathbb{C}$ of polynomials with only real zeros. To show that G_ε belongs to the Laguerre-Pólya class, it is sufficient to show (Dryanov and Rahman, 1999, p. 22) that:

- The zeros of G_ε are all real.
- If $(z_n)_{n\in\mathbb{N}}$ denotes the sequence of real zeros of G_ε, one has $\sum \frac{1}{|z_n|^2} < \infty$.
- G_ε is of order 1.

First, the zeros of G_ε are all real, as demonstrated in Runckel (1969). Second, if $(z_n)_{n\in\mathbb{N}}$ denotes the sequence of real zeros of G_ε, one has $z_n \sim n\pi + \frac{\pi}{2}$ as $n \to \infty$, so that $\sum \frac{1}{|z_n|^2} < \infty$. Third, G_ε is of order 1. Thus, we have that G_ε is indeed in the Laguerre-Pólya class.

This class being stable under differentiation, we get that G'_ε also belongs to the Laguerre-Pólya class. So that the roots of G'_ε are all real, and hence those of f_ε as well.

□

4.A.3.4 Theorem 4.4 in the general case

When $\varepsilon = 0$, we have in the general case the following from Culver (1966):

Let $A \in \mathbb{R}^{d\times d}$. Then A can be represented by a first-order model (4.8) **if and only if** A is not singular and each Jordan block of A corresponding to an eigen value $\lambda < 0$ occurs an even number of time.

We now state and demonstrate the equivalent of this result for second order models (4.7).

Theorem 4.7 (Representable mappings for a Momentum ResNet with linear residual functions – General case)**.** *Let $A \in \mathbb{R}^{d\times d}$.*

If A can be represented by a second-order model (4.7), then each Jordan block of A corresponding to an eigen value $\lambda < \lambda_\varepsilon$ occurs an even number of time.

Reciprocally, if each Jordan block of A corresponding to an eigen value $\lambda \leq \lambda_\varepsilon$ occurs an even number of time, then A can be represented by a second-order model.

Proof. We refer to the arguments from Culver (1966) and use results from Gantmacher (1959) for the proof.

Suppose that A can be represented by a second-order model (4.7). This means that there exists $X \in \mathbb{R}^{d\times d}$ such that $A = f_\varepsilon(X)$. The fact that X is real implies that its Jordan blocks are:

$$\begin{aligned}&(\lambda - z_k)^{a_k},\ z_k \in \mathbb{R}\\ &(\lambda - z_k)^{b_k} \text{ and } (\lambda - \bar{z_k})^{b_k},\ z_k \in \mathbb{C} - \mathbb{R}.\end{aligned}$$

Let $\lambda_k = f_\varepsilon(z_k)$ be an eigenvalue of A such that $\lambda_k < \lambda_\varepsilon$. Necessarily, $z_k \in \mathbb{C} - \mathbb{R}$, and $f'_\varepsilon(z_k) \neq 0$ thanks to Lemma 4.6. We then use Theroem 9 from Gantmacher (1959) (p. 158) to get that the

Jordan blocks of A corresponding to λ_k are

$$(\lambda - f_\varepsilon(z_k))^{b_k} \text{ and } (\lambda - f_\varepsilon(\bar{z_k}))^{b_k}.$$

Since $f_\varepsilon(\bar{z_k}) = f_\varepsilon(z_k) = \lambda_k$, we can conclude that the Jordan blocks of A corresponding $\lambda_k < \lambda_\varepsilon$ occur an even number of time.

Now, suppose that each Jordan block of A corresponding to an eigen value $\lambda \leq \lambda_\varepsilon$ occurs an even number of times. Let λ_k be an eigenvalue of A.

- If $\lambda_k \in \mathbb{C} - \mathbb{R}$ we can write, because f_ε is surjective (proved in Lemma 4.5), $\lambda_k = f_\varepsilon(z_k)$ with $z_k \in \mathbb{C} - \mathbb{R}$. Necessarily, because A is real, the Jordan blocks of A corresponding to λ_k have to be associated to those corresponding to $\bar{\lambda_k}$. In addition, thanks to Lemma 4.6, $f'_\varepsilon(z_k) \neq 0$
- If $\lambda_k < \lambda_\varepsilon$, we can write, because f_ε is surjective, $\lambda_k = f_\varepsilon(z_k) = f_\varepsilon(\bar{z_k})$ with $z_k \in \mathbb{C} - \mathbb{R}$. In addition, $f'_\varepsilon(z_k) \neq 0$.
- If $\lambda_k > \lambda_\varepsilon$, then there exists $z_k \in \mathbb{R}$ such that $\lambda_k = f_\varepsilon(z_k)$ and $f'_\varepsilon(z_k) \neq 0$ because, if x_ε is such that $f_\varepsilon(x_\varepsilon) = \lambda_\varepsilon$, we have that $f'_\varepsilon > 0$ on $]x_\varepsilon, +\infty[$.
- If $\lambda_k = \lambda_\varepsilon$, there exists $z_k \in \mathbb{R}$ such that $\lambda_k = f_\varepsilon(z_k)$. Necessarily, $f'_\varepsilon(z_k) = 0$ but $f''_\varepsilon(z_k) \neq 0$.

This shows that the Jordan blocks of A are necessarily of the form

$$\begin{aligned}
&(\lambda - f_\varepsilon(z_k))^{b_k} \text{ and } (\lambda - f_\varepsilon(\bar{z_k}))^{b_k},\ z_k \in \mathbb{C} - \mathbb{R} \\
&(\lambda - f_\varepsilon(z_k))^{a_k},\ z_k \in \mathbb{R},\ f_\varepsilon(z_k) \neq \lambda_\varepsilon \\
&(\lambda - \lambda_\varepsilon)^{c_k} \text{ and } (\lambda - \lambda_\varepsilon)^{c_k}.
\end{aligned}$$

Let $Y \in \mathbb{R}^{d \times d}$ be such that its Jordan blocks are of the form

$$\begin{aligned}
&(\lambda - z_k)^{b_k} \text{ and } (\lambda - \bar{z_k})^{b_k},\ z_k \in \mathbb{C} - \mathbb{R},\ f'_\varepsilon(z_k) \neq 0 \\
&(\lambda - z_k)^{a_k},\ z_k \in \mathbb{R},\ f_\varepsilon(z_k) \neq \lambda_\varepsilon,\ f'_\varepsilon(z_k) \neq 0 \\
&(\lambda - z_k)^{2c_k},\ z_k \in \mathbb{R},\ f_\varepsilon(z_k) = \lambda_\varepsilon.
\end{aligned}$$

Then again by the use of Theorem 7 from Gantmacher (1959) (p. 158), because if $f_\varepsilon(z_k) = \lambda_\varepsilon$ with $z_k \in \mathbb{R}$, $f''_\varepsilon(z_k) \neq 0$, we have that $f_\varepsilon(Y)$ is similar to A. Thus A writes $A = P^{-1} f_\varepsilon(Y) P = f_\varepsilon(P^{-1}YP)$ with $P \in \mathrm{GL}_d(\mathbb{R})$. Then, $X = P^{-1}YP$ satisfies $X \in \mathbb{R}^{d \times d}$ and $f_\varepsilon(X) = A$. □

4.B Additional theoretical results

4.B.1 On the convergence of the solution of a second order model when $\varepsilon \to \infty$

Proposition 4.8 (Convergence of the solution when $\varepsilon \to +\infty$)**.** *We let x^* (resp. x_ε) be the solution of $\ddot{x} = f(x, \theta)$ (resp. $\ddot{x} + \frac{1}{\varepsilon}\dot{x} = f(x, \theta)$) on $[0, T]$, with initial conditions $x^*(0) = x_\varepsilon(0) = x_0$ and $\dot{x}^*(0) = \dot{x}_\varepsilon(0) = v_0$. Then x_ε converges uniformly to x^* as $\varepsilon \to +\infty$.*

Proof. The equation $\ddot{x} + \frac{1}{\varepsilon}\dot{x} = f(x, \theta)$ with $x_\varepsilon(0) = x_0$, $\dot{x}_\varepsilon(0) = v_0$ writes in phase space (x, v)

$$\begin{cases} \dot{x} &= v, \quad x(0) = x_0 \\ \dot{v} &= f(x, \theta) - \frac{v}{\varepsilon}, \quad v(0) = v_0. \end{cases}$$

It then follows from the Cauchy-Lipschitz Theorem with parameters (Perko, 2013, Theorem 2, Chapter 2) that the solutions of this system are continuous in the parameter $\frac{1}{\varepsilon}$. That is x_ε converges uniformly to x^* as $\varepsilon \to +\infty$.

□

4.B.2 Universality of Momentum ResNets

Proposition 4.9 (When v_0 is free any mapping can be represented)**.** *Consider $h : \mathbb{R}^d \to \mathbb{R}^d$, and the ODE*

$$\ddot{x} + \dot{x} = 0$$
$$(x(0), \dot{x}(0)) = (x_0, \frac{h(x_0) - x_0}{1 - 1/e})$$

Then $\varphi_1(x_0) = h(x_0)$.

Proof. This is because the solution is $\varphi_t(x_0) = x_0 - v_0(e^{-t} - 1)$. □

4.B.3 Non-universality of Momentum ResNets when $v_0 = 0$

Proposition 4.10 (When $v_0 = 0$ there are mappings that cannot be learned if the equation is autonomous.)**.** *When $d = 1$, consider the autonomous ODE*

$$\begin{aligned} \varepsilon\ddot{x} + \dot{x} &= f(x) \\ (x(0), \dot{x}(0)) &= (x_0, 0) \end{aligned} \tag{4.13}$$

If there exists $x_0 \in \mathbb{R}+^*$ such that $h(x_0) \leq -x_0$ and $x_0 \leq h(-x_0)$ then h cannot be represented by (4.13).

This in particular proves that $x \mapsto \lambda x$ for $\lambda \leq -1$ cannot be represented by this ODE with initial conditions $(x_0, 0)$.

Proof. Consider such an x_0 and h. Since $\varphi_1(x_0) = h(x_0) \leq -x_0$, that $\varphi_0(x_0) = x_0$ and that $t \mapsto \varphi_t(x_0)$ is continuous, we know that there exists $t_0 \in [0, 1]$ such that $\varphi_{t_0}(x_0) = -x_0$. We denote $x(t) = \varphi_t(x_0)$, solution of

$$\ddot{x} + \frac{1}{\varepsilon}\dot{x} = f(x)$$

Since $d = 1$, one can write f as a derivative: $f = -E'$. The energy $E_m = \frac{1}{2}\dot{x}^2 + E$ satisfies:

$$\dot{E_m} = -\frac{1}{\varepsilon}\dot{x}^2$$

So that

$$E_m(t_0) - E_m(0) = -\frac{1}{\varepsilon}\int_0^{t_0} \dot{x}^2$$

In other words:

$$\frac{1}{2}v(t_0)^2 + \frac{1}{\varepsilon}\int_0^{t_0} \dot{x}^2 + E(-x_0) = E(x_0)$$

So that $E(-x_0) \leq E(x_0)$ We now apply the exact same argument to the solution starting at $x_1 = -x_0$. Since $x_0 \leq h(-x_0) = h(x_1)$ there exists $t_1 \in [0,1]$ such that $\varphi_{t_1}(x_1) = x_0$. So that:

$$\frac{1}{2}v(t_1)^2 + \frac{1}{\varepsilon}\int_0^{t_1} \dot{x}^2 + E(x_0) = E(-x_0)$$

So that $E(x_0) \leq E(-x_0)$. We get that

$$E(x_0) = E(-x_0)$$

This implies that $\dot{x} = 0$ on $[0, t_0]$, so that the first solution is constant and $x_0 = -x_0$ which is absurd because $x_0 \in \mathbb{R}*$. □

4.B.4 When $v_0 = 0$ there are mappings that can be represented by a second-order model but not by a first-order one.

Proposition 4.11. *There exits f such that the solution of*

$$\ddot{x} + \frac{1}{\varepsilon}\dot{x} = f(x)$$

with initial condition $(x_0, 0)$ at time 1 is

$$x(1) = -x_0 \times \exp(-\frac{1}{2\varepsilon})$$

Proof. Consider the ODE

$$\ddot{x} + \frac{1}{\varepsilon}\dot{x} = (-\pi^2 - \frac{1}{4\varepsilon^2})x \tag{4.14}$$

with initial condition $(x_0, 0)$ The solution of this ODE is

$$x(t) = x_0 e^{-\frac{t}{2\varepsilon}}(\cos(\pi t) + \frac{1}{2\pi\varepsilon}\sin(\pi t))$$

which at time 1 gives:

$$x(1) = -x_0 e^{-\frac{1}{2\varepsilon}}$$

□

4.B.5 Orientation preservation of first-order ODEs

Proposition 4.12 (The homeomorphisms represented by (4.5) are orientation preserving.)**.** *If $K \subset \mathbb{R}^d$ is a compact set and $h : K \to \mathbb{R}^d$ is a homeomorphism represented by (4.5), then h is in the connected component of the identity function on K for the $\|.\|_\infty$ topology.*

We first prove the following:

Lemma 4.13. *Consider $K \subset \mathbb{R}^d$ a compact set. Suppose that $\forall x \in K$, $\Phi_t(x)$ is defined for all $t \in [0,1]$. Then*

$$C = \{\Phi_t(x) \mid x \in K, t \in [0,1]\}$$

is compact as well.

Proof. We consider $(\Phi_{t_n}(x_n))_{n\in\mathbb{N}}$ a sequence in C. Since $K\times[0,1]$ is compact, we can extract sub sequences $(t_{\varphi(n)})_{n\in\mathbb{N}}$, $(x_{\varphi(n)})_{n\in\mathbb{N}}$ that converge respectively to t_0 and x_0. We denote them $(t_n)_{n\in\mathbb{N}}$ and $(x_n)_{n\in\mathbb{N}}$ again for simplicity of the notations. We have that:

$$\|\Phi_{t_n}(x_n)-\Phi_t(x)\|\leq\|\Phi_{t_n}(x_n)-\Phi_{t_n}(x)\|+\|\Phi_{t_n}(x)-\Phi_t(x)\|.$$

Thanks to Gronwall's lemma, we have

$$\|\Phi_{t_n}(x_n)-\Phi_{t_n}(x)\|\leq\|x_n-x\|\exp(kt_n),$$

where k is f's Lipschitz constant. So that $\|\Phi_{t_n}(x_n)-\Phi_{t_n}(x)\|\to 0$ as $n\to\infty$. In addition, it is obvious that $\|\Phi_{t_n}(x)-\Phi_t(x)\|\to 0$ as $n\to\infty$. We conclude that

$$\Phi_{t_n}(x_n)\to\Phi_t(x)\in C,$$

so that C is compact. □

Proof. Let's denote by H the set of homeomorphisms defined on K. The application

$$\Psi:[0,1]\to H$$

defined by

$$\Psi(t)=\Phi_t$$

is continuous. Indeed, we have for any x_0 in $\mathbb{R}^d$ that

$$\|\Phi_{t+\varepsilon}(x_0)-\Phi_t(x_0)\|=\|\int_t^{t+\varepsilon}f(\Phi_s(x_0))ds\|\leq\varepsilon M_f,$$

where M_f bounds the continuous function f on C defined in lemma 4.13. Since M_f does not depend on x_0, we have that

$$\|\Phi_{t+\varepsilon}-\Phi_t\|_\infty\to 0$$

as $\varepsilon\to 0$, which proves that Ψ is continuous. Since $\Psi(0)=Id_K$, we get that $\forall t\in[0,1]$, Φ_t is connected to Id_K. □

4.B.6 On the linear mappings represented by autonomous first order ODEs in dimension 1

Consider the autonomous ODE

$$\dot{x}=f(x),\tag{4.15}$$

Theorem 4.14 (Linearity)**.** *Suppose $d=1$. If (4.15) represents a linear mapping $x\mapsto ax$ at time 1, we have that f is linear.*

Proof. If $a=1$, consider some $x_0\in\mathbb{R}$. Since $\Phi_1(x_0)=x_0=\Phi_0(x_0)$, there exists, by Rolle's Theorem a $t_0\in[0,1]$ such that $\dot{x}(t_0)=0$. Then $f(x(t_0))=0$. But since the constant solution $y=x(t_0)$ then solves $\dot{y}=f(y), y(0)=x(t_0)$, we get by the unicity of the solutions that $x(t_0)=y(0)=x(1)=y(1-t_0)=x_0$. So that $f(x_0)=f(x(t_0))=0$. Since this is true for all x_0, we get that $f=0$. We now consider the case where $a\neq 1$ and $a>0$. Consider some $x_0\in\mathbb{R}^*$. If $f(x_0)=0$, then the solution constant to x_0 solves (4.14), and thus cannot reach ax_0 at time 1 because $a\neq 1$. Thus, $f(x_0)\neq 0$ if $x_0\neq 0$. Second, if the trajectory starting at $x_0\in\mathbb{R}^*$ crosses

0 and $f(0) = 0$, then by the same argument we know that $x_0 = 0$, which is absurd. So that, $\forall x_0 \in \mathbb{R}^*$, $\forall t \in [0,1]$, $f(\Phi_t(x_0)) \neq 0$. We can thus rewrite (4.14) as

$$\frac{\dot{x}}{f(x)} = 1. \tag{4.16}$$

Consider F a primitive of $\frac{1}{f}$. Integrating (4.16), we get

$$F(ax_0) - F(x_0) = \int_0^1 F'(x(t))\dot{x}(t)\mathrm{d}t = 1.$$

In other words, $\forall x \in \mathbb{R}*$:

$$F(ax) = F(x) + 1.$$

We derive this equation and get:

$$af(x) = f(ax).$$

This proves that $f(0) = 0$. We now suppose that $a > 1$. We also have that

$$a^n f(\frac{x}{a^n}) = f(x).$$

But when $n \to \infty$, $f(\frac{x}{a^n}) = \frac{x}{a^n} f'(0) + o(\frac{1}{a^n})$ so that

$$f(x) = f'(0)x$$

and f is linear. The case $a < 1$ treats similarly by changing a^n to a^{-n}. □

4.B.7 There are mappings that are connected to the identity that cannot be represented by a first order autonomous ODE

In bigger dimension, we can exhibit a matrix in $\mathrm{GL}_d^+(\mathbb{R})$ (and hence connected to the identity) that cannot be represented by the autonomous ODE (4.15).

Proposition 4.15 (A non-representable matrix)**.** *Consider the matrix*

$$A = \begin{pmatrix} -1 & 0 \\ 0 & -\lambda \end{pmatrix},$$

where $\lambda > 0$ and $\lambda \neq 1$. Then $A \in GL_2^+(\mathbb{R}) - GL_2(\mathbb{R})^2$ and A cannot be represented by (4.15).

Proof. The fact that $A \in GL_2^+(\mathbb{R}) - GL_2(\mathbb{R})^2$ is because A has two single negative eigenvalues, and because $\det(A) = \lambda > 0$. We consider the point $(0,1)$. At time 1, it has to be in $(0,-\lambda)$. Because the trajectory are continuous, there exists $0 < t_0 < 1$ such that the trajectory is at $(x,0)$ at time t_0, and thus at $(-x,0)$ at time $t_0 + 1$, and again at $(x,0)$ at time $t_0 + 2$. However, the particle is at $(0,\lambda^2)$ at time 2. All of this is true because the equation is autonomous. Now, we showed that trajectories starting at $(0,1)$ and $(0,\lambda^2)$ would intersect at time t_0 at $(x,0)$, which is absurd. Figure 4.11 illustrates the paradox. □

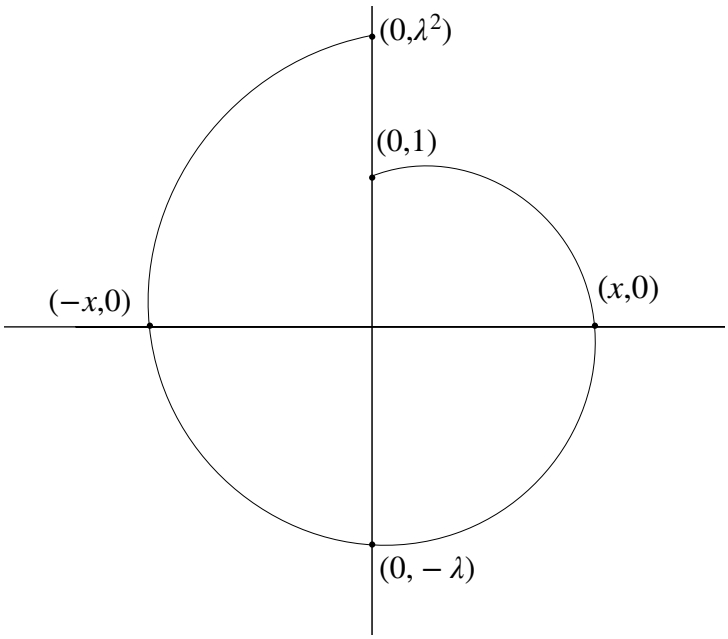

Figure 4.11: Illustration of Proposition 4.15. The points starting at $(0, 1)$ and $(0, \lambda^2)$ are distinct but their associated trajectories would have to intersect in $(x, 0)$, which is impossible.

4.C Exact multiplication

We here present the algorithm from Maclaurin et al. (2015). In their paper, the authors represent γ as a rational number, $\gamma = \frac{n}{d} \in \mathbb{Q}$. The information is lost during the integer division of v_n by d in (4.2). The store this information, it is sufficient to store the remainder r of this integer division. r is stored in an "information buffer" i. To update i, one has to left-shift the bits in i by multiplying it by n before adding r.

Algorithm 1 Exactly reversible multiplication by a ratio, from Maclaurin et al. (2015)

Input: Information buffer i, value c, ratio n/d
$i = i \times d$
$i = i + (c \mod d)$
$c = c \div d$
$c = c \times n$
$c = c + (i \mod n)$
$i = i \div n$
return updated buffer i, updated value c

4.D Experiment details

In all our image experiments, we use Nvidia Tesla V100 GPUs.

For our experiments on CIFAR-10 and 100, we used a batch-size of 128 and we employed SGD with a momentum of 0.9. The training was done over 220 epochs. The initial learning rate was 0.01 and was decayed by a factor 10 at epoch 180. A constant weight decay was set to 5×10^{-4}. Standard inputs preprocessing as proposed in Pytorch (Paszke et al., 2017) was

performed.

For our experiments on ImageNet, we used a batch-size of 256 and we employed SGD with a momentum of 0.9. The training was done over 100 epochs. The initial learning rate was 0.1 and was decayed by a factor 10 every 30 epochs. A constant weight decay was set to 10^{-4}. Standard inputs preprocessing as proposed in Pytorch (Paszke et al., 2017) was performed: normalization, random croping of size 224×224 pixels, random horizontal flip.

For our experiments in the continuous framework, we adapted the code made available by Chen et al. (2018) to work on the CIFAR-10 data set and to solve second order ODEs. We used a batch-size of 128, and used SGD with a momentum of 0.9. The initial learning rate was set to 0.1 and reduced by a factor 10 at iteration 60. The training was done over 120 epochs.

For the learning to optimize experiment, we generate a random Gaussian matrix D of size 16×32. The columns are then normalized to unit variance. We train the networks by stochastic gradient descent for 10000 iterations, with a batch-size of 1000 and a learning rate of 0.001. The samples y_q are generated as follows: we first sample a random Gaussian vector $\tilde{y}_q$, and then we use $y_q = \frac{\tilde{y}_q}{\|D^\top \tilde{y}_q\|_\infty}$, which ensures that every sample verify $\|D^\top y_q\|_\infty = 1$. This way, we know that the solution x^* is zero if and only if $\lambda \geq 1$. The regularization is set to $\lambda = 0.1$.

4.E Backpropagation for Momentum ResNets

In order to backpropagate the gradient of some loss in a Momentum ResNet, we need to formulate an explicit version of (4.2). Indeed, (4.2) writes explicitly

$$\begin{aligned} v_{n+1} &= \gamma v_n + (1-\gamma) f(x_n, \theta_n) \\ x_{n+1} &= x_n + (\gamma v_n + (1-\gamma) f(x_n, \theta_n)). \end{aligned} \tag{4.17}$$

Writing $z = (x, v)$, the backpropagation for Momentum ResNets then writes, for some loss L

$$\nabla_{z_{k-1}} L = \begin{bmatrix} I + (1-\gamma)\partial_x f(x_{k-1}, \theta_{k-1}) & \gamma I \\ (1-\gamma)\partial_x f(x_{k-1}, \theta_{k-1}) & \gamma I \end{bmatrix}^T \nabla_{z_k} L$$

$$\nabla_{\theta_{k-1}} L = (1-\gamma) \begin{bmatrix} \partial_\theta f(x_{k-1}, \theta_{k-1}) \\ \partial_\theta f(x_{k-1}, \theta_{k-1}) \end{bmatrix}^T \nabla_{z_k} L.$$

We implement these formula to obtain a custom Jacobian-vector product in Pytorch.

4.F Additional figures

4.F.1 Learning curves on CIFAR-10

We here show the learning curves when training a ResNet-101 and a Momentum ResNet-101 on CIFAR-10.

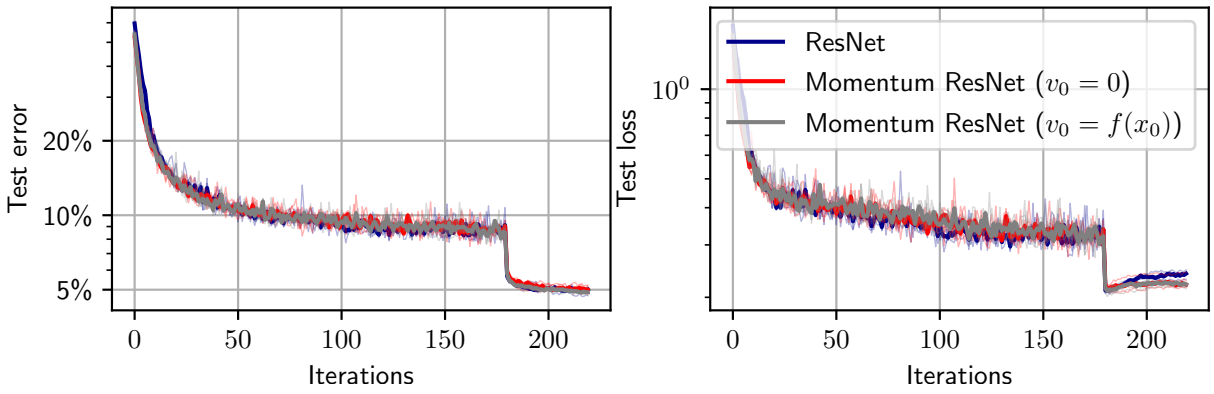

Figure 4.12: Test error and test loss as a function of depth on CIFAR-10 with a ResNet-101 and two Momentum ResNets-101.

5

Sinkformers: Transformers with Doubly Stochastic Attention

Attention based models such as Transformers involve pairwise interactions between data points, modeled with a learnable attention matrix. Importantly, this attention matrix is normalized with the SoftMax operator, which makes it row-wise stochastic. In this chapter, we propose instead to use Sinkhorn's algorithm to make attention matrices doubly stochastic. We call the resulting model a Sinkformer. We show that the row-wise stochastic attention matrices in classical Transformers get close to doubly stochastic matrices as the number of epochs increases, justifying the use of Sinkhorn normalization as an informative prior. On the theoretical side, we show that, unlike the SoftMax operation, this normalization makes it possible to understand the iterations of self-attention modules as a discretized gradient-flow for the Wasserstein metric. We also show in the infinite number of samples limit that, when rescaling both attention matrices and depth, Sinkformers operate a heat diffusion. On the experimental side, we show that Sinkformers enhance model accuracy in vision and natural language processing tasks. In particular, on 3D shapes classification, Sinkformers lead to a significant improvement.

Contents

5.1 Introduction

The Transformer (Vaswani et al., 2017), an architecture that relies entirely on attention mechanisms (Bahdanau et al., 2014a), has achieved state of the art empirical success in natural language processing (NLP) (Brown et al., 2020; Radford et al., 2019; Wolf et al., 2019) as well as in computer vision (Dosovitskiy et al., 2020; Zhao et al., 2020; Zhai et al., 2021; Lee et al., 2019a). As the key building block of the Transformer, the self-attention mechanism takes the following residual form (Yun et al., 2019) given a n-sequence $(x_1, x_2, ..., x_n)$, embedded in dimension d:

$$x_i \leftarrow x_i + \sum_{j=1}^{n} K^1_{i,j} W_V x_j, \tag{5.1}$$

where $K^1 \coloneqq \texttt{SoftMax}(C)$ with $C_{i,j} \coloneqq (W_Q x_i)^\top W_K x_j = x_i^\top W_Q^\top W_K x_j$. Here, $W_Q, W_K \in \mathbb{R}^{m\times d}$ and $W_V \in \mathbb{R}^{d\times d}$ are the query, key and value matrices. The SoftMax operator can be seen as a normalization of the matrix $K^0 \coloneqq \exp(C)$ as follows: $K^1_{ij} \coloneqq K^0_{ij} / \sum_{l=1}^n K^0_{il}$ for all i and j. Importantly, the matrix K^1 is row-wise stochastic: its rows all sum to 1.

In this work, we propose to take the normalization process further by successively normalizing the rows and columns of K^0. This process is known to provably converge to a doubly stochastic matrix (i.e., whose rows and columns both sum to 1) and is called Sinkhorn's algorithm (Sinkhorn, 1964; Cuturi, 2013; Peyré et al., 2019). We denote the resulting doubly stochastic matrix K^∞. Intuitively, such a normalization relies on a democratic principle where all points are matched one to another with different degrees of intensity, so that more interactions are considered than with the SoftMax normalization, as shown in Figure 5.1.

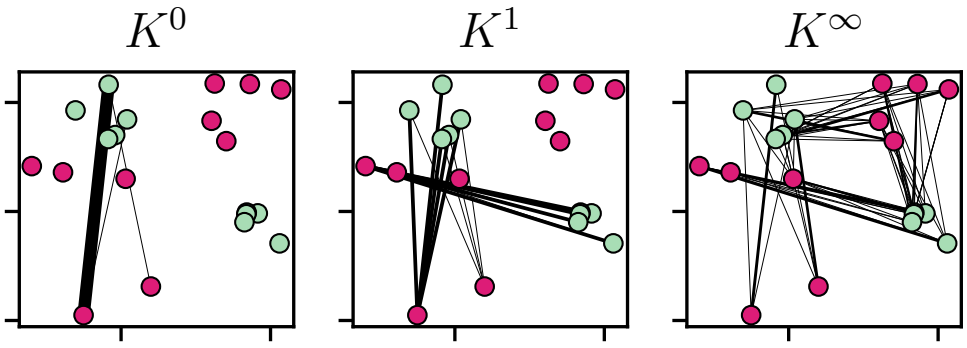

Figure 5.1: **Illustration of the different normalizations of attention matrices.** We form two point clouds $(W_Q x_i)_{1\leq i\leq 10}$ (green) and $(W_K x_j)_{1\leq i\leq 10}$ (red). For $k \in \{0, 1, \infty\}$, the width of the line connecting x_i to x_j is $K^k_{i,j}$. We only display connections with $K^k_{i,j} \geq 10^{-12}$. For K^0, one interaction dominates. For K^1 (SoftMax), one cluster is ignored. For K^∞ (Sinkhorn), all points are involved in an interaction.

We call our Transformer variant where the SoftMax is replaced by Sinkhorn a **Sinkformer**. Since Sinkhorn's first iteration coincides exactly with the SoftMax, Sinkformers include Transformers as a special case. Our modification is differentiable, easy to implement using deep learning libraries, and can be executed on GPUs for fast computation. Because the set of row-wise stochastic matrices contains the set of doubly stochastic matrices, the use of doubly stochastic matrices can be interpreted as a prior. On the experimental side, we confirm that doubly stochastic attention leads to better accuracy in several learning tasks. On the theoretical side, doubly stochastic matrices also give a better understanding of the mathematical properties of self-attention maps.

To summarize, we make the following contributions.

- We show empirically that row-wise stochastic matrices seem to converge to doubly stochastic matrices during the learning process in several classical Transformers (Figure 5.2). Motivated by this finding, we then introduce the Sinkformer, an extension of the Transformer in which the SoftMax is replaced by the output of Sinkhorn's algorithm. In practice, our model is parametrized by the number of iterations in the algorithm, therefore interpolating between the Transformer and the Sinkformer.
- On the theoretical side, we show that Transformers and Sinkformers can be viewed as models acting on discrete distributions, and we show under a symmetry assumption that Sinkformers can be seen in the infinite depth limit as a Wasserstein gradient flow for an energy minimization (Proposition 5.3). We also show that the classical Transformer with the SoftMax operator cannot be interpreted as such a flow (Proposition 5.4). To the best of our knowledge, this is the first time such a connection is established. We also prove that in the infinite number of particles limit (when n goes to infinity), the iterations of Sinkformers converge to the heat equation (Theorem 5.5), while the corresponding equation for Transformers is nonlinear and nonlocal (Proposition 5.6).
- On the experimental side, we show that Sinkformers lead to a significant accuracy gain compared to Transformers on the ModelNet 40 3D shapes classification task. We then demonstrate better performance of Sinkformers on the NLP IMDb dataset for sentiment analysis and IWSLT'14 German to English neural machine translation tasks. Sinkformers also achieve a better accuracy than Vision Transformers on image classification tasks. Therefore, the proposed method is capable of enhancing the performance of transformers in a wide range of applications.

5.2 Background and related works

Transformers. Proposed by Vaswani et al. (2017), the Transformer is a fully attention-based architecture. Originally designed to process sequences for natural language processing (NLP), many variants have since been developed such as Vision Transformers (Dosovitskiy et al., 2020; Zhai et al., 2021), Set Transformers (Lee et al., 2019a) or Point Cloud Transformers (Zhao et al., 2020). The Transformer and its variants are based on an encoder-decoder structure, where the decoder can have a more or less complex form. The encoder is fully *self*-attention based. After embedding and concatenating with positional encoding the original input sequence, the encoder uses a series of residual blocks that iterates relation (5.1) followed by a feed forward neural network applied to each x_i independently. In its most complex form such as in neural machine translation, the decoder combines a self-attention based mechanisms and a *cross* attention one, meaning that it is given access to the encoder via another multi-head attention block.

Sinkhorn and Attention. To the best of our knowledge, using Sinkhorn's algorithm in Transformers has been done once in a different context (Tay et al., 2020). The authors propose to learn efficient and sparse attention using a differentiable algorithm for sorting and rearranging elements in the input sequence. For this purpose, they introduce a sorting network to generate a doubly-stochastic matrix (that can be seen as a relaxed version of a permutation matrix) and use it to sort the sequence in a differentiable fashion. Mialon et al. (2021a) propose an embedding for sets of features in $\mathbb{R}^d$ based on Sinkhorn's algorithm, by using the regularized optimal transport plan between data points and a reference set. Niculae et al. (2018) use doubly stochastic attention matrices in LSTM-based encoder-decoder networks but they use Frank-Wolfe or active set methods to compute the attention matrix. None of these works use Sinkhorn on self-attention maps in Transformers and provide its theoretical analysis, as we do.

Impact of bi-normalization. Theoretical properties of kernels $\mathcal{K}$, which attention is an instance of, can also be studied through the operator $f \mapsto f - \mathcal{K}f$. Bi-normalization of kernels over manifolds have already been studied in the literature, on uniform measures (Singer, 2006), weighted measures (Hein et al., 2007) and in a more general setup with associated diffusion operators (Ting et al., 2011). Milanfar (2013) proposes to approximate smoothing operators by doubly stochastic matrices using Sinkhorn's updates, leading to better performance in data analysis and signal processing. Importantly, the works of Marshall and Coifman (2019) and Wormell and Reich (2021) exactly introduce a normalization that is based on Sinkhorn's algorithm. They prove that this method models a Langevin diffusion and leads to the approximation of a symmetric operator. They also show that convergence to this operator is faster with Sinkhorn normalization than with the SoftMax normalization. In section 5.5, we adopt a similar point of view with a parametrized cost and show that different normalizations result in different partial differential equations (PDEs) in the infinite number of particles limit.

Infinite depth limit. Studying deep residual neural networks (ResNets) (He et al., 2016a) in the infinitesimal step-size regime (or infinite depth limit) has recently emerged as a new framework for analyzing their theoretical properties. The ResNet equation

$$x_i \leftarrow x_i + T(x_i) \tag{5.2}$$

can indeed be seen as a discretized Euler scheme with unit step size of the ordinary differential equation (ODE) $\dot{x_i} = T(x_i)$ (Weinan, 2017b; Chen et al., 2018; Teh et al., 2019; Sun et al., 2018; Weinan et al., 2019; Lu et al., 2018; Ruthotto and Haber, 2019; Sander et al., 2021). In section 5.4, we adopt this point of view on residual attention layers in order to get a better theoretical understanding of attention mechanisms. This is justified by the fact that, for instance, GPT-3 (Brown et al., 2020) has 96 layers.

Neural networks on measures. The self-attention mechanism (5.1) acts on sets $\{x_i\}_i$ where the ordering of the elements does not matter. An equivalent way to model such invariant architectures is to consider them as acting on probability measures or point clouds of varying cardinality (De Bie et al., 2019; Vuckovic et al., 2021; Zweig and Bruna, 2021). Specifically, a collection of points $(x_i)_{1\leq i\leq n}$, where $x_i \in \mathbb{R}^d$, can also be seen as a discrete measure on $\mathbb{R}^d$: $\mu \coloneqq \frac{1}{n}\sum_{i=1}^n \delta_{x_i} \in \mathcal{M}(\mathbb{R}^d)$, where $\mathcal{M}(\mathbb{R}^d)$ is the set of probability measures on $\mathbb{R}^d$. A map T_μ then acts on μ through $F(\mu) \coloneqq \frac{1}{n}\sum_{i=1}^n \delta_{T_\mu(x_i)}$. One notable interest of such a point of view is to consider the evolution of non ordered sets of points. Another is to consider the mean field (or large sample) limit, that is when $n \to \infty$, to conduct theoretical analysis (Zweig and Bruna, 2021) as when analyzing the SGD properties in the mean-field limit (Song et al., 2018).

5.3 Sinkformers

We now introduce Sinkformers, a modification of any Transformer by replacing the SoftMax operator in the attention modules by Sinkhorn's algorithm.

Attention matrices during training. In Transformers, attention matrices are row-wise stochastic. A natural question is how the sum over columns evolve during training. On 3 different models and 3 different learning tasks, we calculated the sum over columns of attention matrices in Transformers. We find out that the learning process makes the attention matrices more and more doubly stochastic, as shown in Figure 5.2.

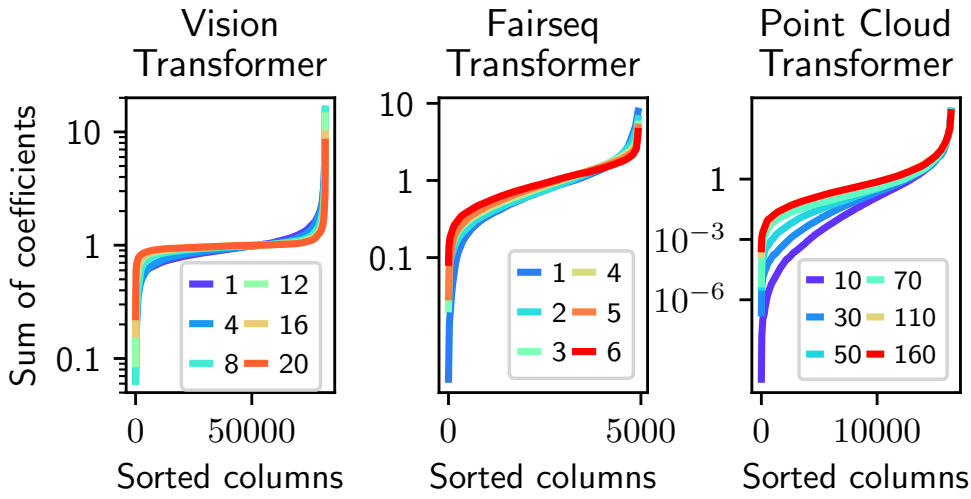

Figure 5.2: **Sum over columns** of attention matrices at different training epochs (color) when training, from left to right, a ViT on MNIST (section 5.6.4), a `fairseq` Transformer on IWSLT'14 (section 5.6.3), and a Point Cloud Transformer on Model Net 40 (section 5.6.1). **The majority of columns naturally sum closely to** 1**.**

Thus, row-wise stochastic attention matrices seem to approach doubly stochastic matrices during the learning process in classical Transformers. Therefore, it seems natural to impose double stochasticity as a prior and study theoretically and experimentally the resulting model. A process to obtain such matrices which extends the SoftMax is Sinkhorn's algorithm.

Sinkhorn's algorithm. Given a matrix $C \in \mathbb{R}^{n\times n}$, and denoting $K^0 \in \mathbb{R}^{n\times n}$ such that $K^0 = \exp(C)$, Sinkhorn's algorithm (Sinkhorn, 1964; Cuturi, 2013; Peyré et al., 2019) iterates, starting from K^0:

$$K^{l+1} = \begin{cases} N_R(K^l) & \text{if } l \text{ is even} \\ N_C(K^l) & \text{if } l \text{ is odd,} \end{cases} \tag{5.3}$$

where N_R and N_C correspond to row-wise and column-wise normalizations: $(N_R(K))_{i,j} \coloneqq \frac{K_{i,j}}{\sum_{l=1}^n K_{i,l}}$ and $(N_C(K))_{i,j} \coloneqq \frac{K_{i,j}}{\sum_{l=1}^n K_{l,j}}$. We denote the resulting scaled matrix limit $K^\infty \coloneqq \texttt{Sinkhorn}(C)$. Note that it is doubly stochastic in the sense that $K^\infty \mathbb{1}_n = \mathbb{1}_n$ and ${K^\infty}^\top \mathbb{1}_n = \mathbb{1}_n$. The operations in (5.3) are perfectly suited for being executed on GPUs (Charlier et al., 2021; Cuturi, 2013).

Sinkformers. For simplicity, we consider a one head attention block that iterates equation (5.1). Note that $K^1 \coloneqq \texttt{SoftMax}(C)$ is precisely the output of Sinkhorn's algorithm (5.3) after 1 iteration. In this chapter, we propose to take Sinkhorn's algorithm several steps further

until it approximately converges to a doubly stochastic matrix K^∞. This process can be easily implemented in practice, simply by plugging Sinkhorn's algorithm into self-attention modules in existing architectures, without changing the overall structure of the network. We call the resulting drop-in replacement of a Transformer a Sinkformer. It iterates

$$x_i \leftarrow x_i + \sum_{j=1}^{n} K^\infty_{i,j} W_V x_j. \tag{5.4}$$

In the next two sections 5.4 and 5.5, we investigate the theoretical properties of Sinkformers. We exhibit connections with energy minimization in the space of measures and the heat equation, thereby proposing a new framework for understanding attention mechanisms. All our experiments are described in Section 5.6 and show the benefits of using Sinkformers in a wide variety of applications.

Computational cost and differentiation. Turning a Transformer into a Sinkformer simply relies on replacing the SoftMax by Sinkhorn, i.e., substituting K^1 with K^∞. In practice, we use a finite number of Sinkhorn iterations and therefore use K^l, where l is large enough so that K^l is almost doubly stochastic. Doing l iterations of Sinkhorn takes l times longer than the SoftMax. However, this is not a problem in practice because Sinkhorn is not the main computational bottleneck and because only a few iterations of Sinkhorn are sufficient (typically 3 to 5) to converge to a doubly stochastic matrix. As a result, the practical training time of Sinkformers is comparable to regular Transformers, as detailed in our experiments.

Sinkhorn is perfectly suited for backpropagation (automatic differentiation), by differentiating through the operations of (5.3). The Jacobian of an optimization problem solution can also be computed using the implicit function theorem (Griewank and Walther, 2008b; Krantz and Parks, 2012; Blondel et al., 2021) instead of backpropagation if the number of iterations becomes a memory bottleneck. Together with Sinkhorn, implicit differentiation has been used by Luise et al. (2018) and Cuturi et al. (2020).

Invariance to the cost function. Recall that in practice one has $C_{i,j} = (W_Q x_i)^\top W_K x_j$. An important aspect of Sinkformers is that their output is unchanged if the cost is modified with non interacting terms, as the next proposition shows.

Proposition 5.1. *Let $C \in \mathbb{R}^{n\times n}$. Consider, for $(f, g) \in \mathbb{R}^n \times \mathbb{R}^n$ the modified cost function $\tilde{C}_{i,j} \coloneqq C_{i,j} + f_i + g_j$. Then $\texttt{Sinkhorn}(C) = \texttt{Sinkhorn}(\tilde{C})$.*

A proof is available in Appendix 5.A. A consequence of this result is that one can consider the cost $\tilde{C}_{i,j} \coloneqq -\frac{1}{2}\|W_Q x_i - W_K x_j\|^2$ instead of $C_{i,j} = (W_Q x_i)^\top W_K x_j$, without affecting K^∞. A Transformer using the cost $\tilde{C}$ is referred to as L2 self-attention, and is Lipschitz under some assumptions (Kim et al., 2021) and can therefore be used as an invertible model (Behrmann et al., 2019). For instance, we use $\tilde{C}$ in Proposition 5.6.

5.4 Attention and gradient flows

In this section, we make a parallel between self-attention modules in Sinkformers and gradient flows in the space of measures. We denote $\mathcal{M}(\mathbb{R}^d)$ the probability measures on $\mathbb{R}^d$ and $\mathcal{C}(\mathbb{R}^d)$ the continuous functions on $\mathbb{R}^d$. We denote ∇ the gradient operator, div the divergence, and Δ the Laplacian, that is $\Delta = \mathrm{div}(\nabla)$.

Residual maps for attention. We consider a one-head attention block operating with different normalizations. We consider the continuous counterparts of the attention matrices seen in the previous section. We denote $c(x, x') \coloneqq (W_Q x)^\top W_K x'$ and $k^0 \coloneqq \exp(c)$. For some measure $\mu \in \mathcal{M}(\mathbb{R}^d)$, we define the $\texttt{SoftMax}$ operator on the cost c by $k^1(x, x') = \texttt{SoftMax}(c)(x, x') \coloneqq \frac{k^0(x,x')}{\int k^0(x,y)d\mu(y)}$. Similarly, we define Sinkhorn's algorithm as the following iterations, starting from $k^0 = \exp(c)$:

$$k^{l+1}(x, x') = \begin{cases} \frac{k^l(x,x')}{\int k^l(x,y)d\mu(y)} & \text{if } l \text{ is even} \\ \frac{k^l(x,x')}{\int k^l(y,x)d\mu(y)} & \text{if } l \text{ is odd.} \end{cases} \tag{5.5}$$

We denote $k^\infty \coloneqq \texttt{Sinkhorn}(c)$ the resulting limit. Note that if μ is a discrete measure supported on a n sequence of particles $(x_1, x_2, ..., x_n)$, $\mu = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$, then for all (i, j), $k^0(x_i, x_j) = K^0_{i,j}$, $k^1(x_i, x_j) = K^1_{i,j}$ and $k^\infty(x_i, x_j) = K^\infty_{i,j}$, so that k^0, k^1 and k^∞ are indeed the continuous equivalent of the matrices K^0, K^1 and K^∞ respectively.

Infinitesimal step-size regime. In order to better understand the theoretical properties of attention matrices in Transformers and Sinkformers, we omit the feed forward neural networks acting after each attention block. We consider a succession of attention blocks with tied weights between layers and study the infinite depth limit where the output is given by solving a neural ODE (Chen et al., 2018). In this framework, iterating the Transformer equation (5.1), the ResNet equation (5.2) and the Sinkformer equation (5.4) corresponds to a Euler discretization with step-size 1 of the ODEs

$$\dot{x}_i = T_\mu(x_i) \text{ for all } i, \tag{5.6}$$

where $x_i(t)$ is the position of x_i at time t. For an arbitrary measure $\mu \in \mathcal{M}(\mathbb{R}^d)$, these ODEs can be equivalently written as a continuity equation (Renardy and Rogers, 2006)

$$\partial_t \mu + \operatorname{div}(\mu T_\mu) = 0. \tag{5.7}$$

When T_μ is defined by the ResNet equation (5.2), $T_\mu = T$ does not depend on μ. It defines an advection equation where the particles do not interact and evolve independently. When T_μ is defined by the Transformer equation (5.1) or Sinkformer equation (5.4), T_μ has a dependency in μ and the particles interact: the local vector field depends on the position of the other particles. More precisely we have in this case $T^1_\mu(x) = \int k^1(x, x')W_V x' d\mu(x')$ for the Transformer and $T^\infty_\mu(x) = \int k^\infty(x, x')W_V x' d\mu(x')$ for the Sinkformer. It is easily seen that when μ is discrete we recover the operators in equation (5.1) and (5.4).

Wasserstein gradient flows. A particular case of equation (5.7) is when T_μ is a gradient with respect to the Wasserstein metric W_2. Let $\mathcal{F}$ be a function on $\mathcal{M}(\mathbb{R}^d)$. As is standard, we suppose that $\mathcal{F}$ admits a first variation at all μ: there exists a function $\frac{\delta \mathcal{F}}{\delta \mu}(\mu)$ such that $\frac{d}{d\varepsilon}\mathcal{F}(\mu + \varepsilon\rho)_{|\varepsilon=0} = \int \frac{\delta \mathcal{F}}{\delta \mu}(\mu)d\rho$ for every perturbation ρ (Santambrogio, 2017). The Wasserstein gradient of $\mathcal{F}$ at μ is then $\nabla_W \mathcal{F}(\mu) \coloneqq \nabla(\frac{\delta \mathcal{F}}{\delta \mu}(\mu))$. The minimization of $\mathcal{F}$ on the space of measures corresponds to the PDE (5.7) with $T_\mu = -\nabla_W \mathcal{F}(\mu)$. This PDE can be interpreted as ruling the evolution of the measure μ of particles initially distributed according to some measure μ_0, for which the positions $x(t)$ follow the flow $\dot{x} = -\nabla_W \mathcal{F}(\mu)(x)$, that minimizes the global energy $\mathcal{F}$. It corresponds to a steepest descent in Wasserstein space (Jordan et al., 1998). In Proposition 5.3, we show in the symmetric kernel case that Sinkformers correspond to a Wasserstein gradient flow for some functional $\mathcal{F}^\infty$, while Transformers do not.

Particular case. An example is when T_μ does not depend on μ and writes $T_\mu = -\nabla E$ where $E : \mathbb{R}^d \to \mathbb{R}$. Under regularity assumptions, a solution of (5.7) then converges to a local minimum of E. This fits in the implicit deep learning framework (Bai et al., 2019), where a neural network is seen as solving an optimization problem. A typical benefit of implicit models is that the iterates x_i do not need to be stored during the forward pass of the network because gradients can be calculated using the implicit function theorem: it bypasses the memory storage issue of GPUs (Wang et al., 2018; Peng et al., 2017; Zhu et al., 2017) during automatic differentiation. Another application is to consider neural architectures that include an argmin layer, for which the output is also formulated as the solution of a nested optimization problem (Agrawal et al., 2019; Gould et al., 2016, 2019).

Flows for attention. Our goal is to determine the PDEs (5.7) defined by the proposed attention maps. We consider the symmetric case, summarized by the following assumption:

Assumption 5.2. ${W_K}^\top W_Q = {W_Q}^\top W_K = -W_V$

Assumption 5.2 means we consider symmetric kernels (by imposing ${W_K}^\top W_Q = {W_Q}^\top W_K$), and that when differentiating $x \mapsto \exp(c(x, x'))$, we obtain $-\exp(c)W_V$. We show that, under this assumption, the PDEs defined by k^0 and k^∞ correspond to Wasserstein gradient flows, whereas it is not the case for k^1. A particular case of imposing ${W_K}^\top W_Q = {W_Q}^\top W_K$ is when $W_Q = W_K$. This equality setting is studied by Kim et al. (2021), where the authors show that it leads to similar performance for Transformers. Since imposing ${W_K}^\top W_Q = {W_Q}^\top W_K$ is less restrictive, it seems to be a natural assumption. Imposing $W_Q^\top W_K = -W_V$ is more restrictive, and we detail the expressions for the PDEs associated to k^0, k^1, k^∞ without this assumption in Appendix 5.A.

We have the following result.

Proposition 5.3 (PDEs associated to k^0, k^1, k^∞)**.** *Suppose Assumption 5.2. Let $\mathcal{F}^0$ and $\mathcal{F}^\infty : \mathcal{M}(\mathbb{R}^d) \to \mathbb{R}$ be such that $\mathcal{F}^0(\mu) \coloneqq \frac{1}{2}\int k^0 d(\mu \otimes \mu)$ and $\mathcal{F}^\infty(\mu) \coloneqq -\frac{1}{2}\int k^\infty \log(\frac{k^\infty}{k^0}) d(\mu \otimes \mu)$. Then k^0, k^1 and k^∞ respectively generate the PDEs $\frac{\partial \mu}{\partial t} + \operatorname{div}(\mu T_\mu^k) = 0$ with $T_\mu^0 \coloneqq -\nabla_W \mathcal{F}^0(\mu)$, $T_\mu^1 \coloneqq -\nabla[\log(\int k^0(\cdot, x')d\mu(x'))]$ and $T_\mu^\infty \coloneqq -\nabla_W \mathcal{F}^\infty(\mu)$.*

A proof is given in Appendix 5.A. Proposition 5.3 shows that k^0 and k^∞ correspond to Wasserstein gradient flows. In addition, the PDE defined by k^1 does not correspond to such a flow. More precisely, we have the following result.

Proposition 5.4 (The SoftMax normalization does not correspond to a gradient flow)**.** *One has that $T_\mu^1 = -\nabla[\log(\int k^0(\cdot, x')d\mu(x'))]$ is not a Wasserstein gradient.*

A proof is given in Appendix 5.A, based on the lack of symmetry of T_μ^1. As a consequence of these results, we believe this variational formulation of attention mechanisms for Sinkformers (Proposition 5.3) provides a perspective for analyzing the theoretical properties of attention-based mechanisms in light of Wasserstein gradient flow theory (Santambrogio, 2017). Moreover, it makes it possible to interpret Sinkformers as argmin layers, which is promising in terms of theoretical and experimental investigations, and which is not possible for Transformers, according to Proposition 5.4.

Our results are complementary to the one of Dong et al. (2021), where the authors show that, **with no skip connections** and without the feed forward neural network acting after each attention block, the output of a Transformer converges doubly exponentially with depth to a rank-1 matrix. On the contrary, we propose a complementary analysis by taking skip-connections into account, as is standard in Transformers. Precisely because we consider such connections, we

end up with very different behaviors. Indeed, as shown in the next section, our analysis reveals that the relative signs for W_K, W_Q and W_V imply very different behavior, such as aggregation or diffusion. The dynamics obtained when considering skip connections are therefore richer than a rank collapse phenomenon.

5.5 Attention and diffusion

In this section, we use the same notations as in section 5.4. We consider the mean-field limit, where the measure μ has a density with respect to the Lebesgue measure. We are interested in how the density of particles evolves for an infinite depth self-attention network with tied weights between layers. We consider Assumption 5.2 and suppose that $W_K^\top W_Q$ is positive semi-definite. For a bandwidth $\varepsilon > 0$, let $k_\varepsilon^\infty = \texttt{Sinkhorn}(c/\varepsilon)$, that is the attention kernel for the Sinkformer with the cost c/ε. The mapping $T_{\mu,\varepsilon}^\infty : x \mapsto \frac{1}{\varepsilon}\int k_\varepsilon^\infty(x,x')W_V x' d\mu(x')$ corresponds to the continuous version of the Sinkformer where we re-scale $W_Q W_K^T = -W_V$ by ε. To better understand the dynamics of attention, we study the asymptotic regime in which the bandwidth $\varepsilon \to 0$. In this regime, one can show that $\forall x \in \mathbb{R}^d$, $\varepsilon T_{\mu,\varepsilon}^\infty(x) \to W_V x$ (details in Appendix 5.A). Thus, to go beyond first order, we study the modified map $\overline{T}_{\mu,\varepsilon}^\infty = T_{\mu,\varepsilon}^\infty - \frac{1}{\varepsilon}W_V$. A natural question is the limit of this quantity when $\varepsilon \to 0$, and what the PDE defined by this limit is. We have the following theorem.

Theorem 5.5 (Sinkformer's PDE)**.** *Let $\mu \in \mathcal{M}(\mathbb{R}^d)$. Suppose that μ is supported on a compact set and has a density $\rho \in \mathcal{C}^3(\mathbb{R}^d)$. Suppose assumption 5.2 and that $W_K^\top W_Q$ is positive semi-definite. Then one has in L^2 norm as $\varepsilon \to 0$,*

$$\overline{T}_{\mu,\varepsilon}^\infty \to \overline{T}_{\mu,0}^\infty := -\frac{\nabla \rho}{\rho}.$$

In this limit, the PDE $\partial_t \rho + \operatorname{div}(\rho \overline{T}_{\mu,0}^\infty) = 0$ rewrites

$$\partial_t \rho = \Delta \rho. \tag{5.8}$$

A proof is available in Appendix 5.A, making use of Theorem 1 from Marshall and Coifman (2019). We recover in Equation (5.8) the well-known **heat equation**.

We want to compare this result with the one obtained with the SoftMax normalization. In order to carry a similar analysis, we make use of a Laplace expansion result (Tierney et al., 1989; Singer, 2006). However, the kernel $k_\varepsilon^1 = \texttt{SoftMax}(c/\varepsilon)$ is not suited for using Laplace method because it does not always have a limit when $\varepsilon \to 0$. Thus, we consider the modified cost as in Proposition 5.1, $\tilde{c}(x,x') = -\frac{\|W_Q x - W_K x'\|^2}{2}$. The kernel $\tilde{k}_\varepsilon^1 = \texttt{SoftMax}(\tilde{c}/\varepsilon)$, for which we can now apply Laplace expansion result, then corresponds to the L2 self-attention formulation (Kim et al., 2021). Note that thanks to Proposition 5.1, $\tilde{k}_\varepsilon^\infty = k_\varepsilon^\infty$: Sinkorn's algorithm will have the same output for both costs. To simplify the expressions derived, we assume that W_Q and W_K are in $\mathbb{R}^{d\times d}$ and are invertible. Similarly to the analysis conducted for Sinkformers, we consider the mapping $T_{\mu,\varepsilon}^1 : x \mapsto \frac{1}{\varepsilon}\int \tilde{k}_\varepsilon^1(x,x')W_V x' d\mu(x')$. When $\varepsilon \to 0$, we show that $\forall x \in \mathbb{R}^d$, $\varepsilon T_{\mu,\varepsilon}^1(x) \to -W_Q^\top W_Q x$ (details in Appendix 5.A). Thus, we consider $\overline{T}_{\mu,\varepsilon}^1 = T_{\mu,\varepsilon}^1 + \frac{1}{\varepsilon}\mathrm{W_Q^\top W_Q}$. We have the following result.

Proposition 5.6 (Transformer's PDE)**.** *Let $\mu \in \mathcal{M}(\mathbb{R}^d)$. Suppose that μ is supported on a compact set and has a density $\rho \in \mathcal{C}^1(\mathbb{R}^d)$. Suppose assumption 5.2 and that W_Q and W_K are in $\mathbb{R}^{d\times d}$ and are invertible. Then one has $\forall x \in \mathbb{R}^d$,*

$$\overline{T}_{\mu,\varepsilon}^1(x) \to \overline{T}_{\mu,0}^1(x) := -W_Q^\top W_K^{-1}\frac{\nabla \rho}{\rho}(W_K^{-1}W_Q x).$$

In this limit, the PDE $\partial_t \rho + \mathrm{div}(\rho \overline{T}^1_{\mu,0}) = 0$ *rewrites*

$$\partial_t \rho = \mathrm{div}(W_Q^\top W_K^{-1} \frac{\nabla \rho}{\rho}(W_K^{-1} W_Q \cdot)\rho) \tag{5.9}$$

A proof is given in Appendix 5.A. While equation (5.8) corresponds to the heat equation, equation (5.9) is different. First, it is nonlinear in ρ. Second, it is nonlocal since the evolution of the density at x depends on the value of this density at location $W_K^{-1} W_Q x$. Note that the linear and local aspect of Sinkformer's PDE on the one hand, and the nonlinear and nonlocal aspect of Transformer's PDE on the other hand, remain true without assuming ${W_Q}^\top W_K = -W_V$ (details in Appendix 5.A).

5.6 Experiments

We now demonstrate the applicability of Sinkformers on a large variety of experiments with different modalities. We use Pytorch (Paszke et al., 2017) and Nvidia Tesla V100 GPUs. Our code is open-sourced and is available at this address: `https://github.com/michaelsdr/sinkformers`. All the experimental details are given in Appendix 5.C.

Practical implementation. In all our experiments, we use existing Transformer architectures and modify the SoftMax operator in attention modules with Sinkhorn's algorithm, which we implement in log domain for stability (details in Appendix 5.B).

5.6.1 ModelNet 40 classification

The ModelNet 40 dataset (Wu et al., 2015) is composed of 40 popular object categories in 3D. Transformers for point clouds and sets have been applied to the ModelNet 40 classification in several works, such as Set Transformers (Lee et al., 2019a) or Point Cloud Transformers (Guo et al., 2021).

Set Sinkformers. Set Transformers (Lee et al., 2019a) also have an encoder decoder structure with different possibilities for defining attention-based set operations. We propose to focus on the architecture that uses *Induced Self Attention Block* (ISAB), which bypasses the quadratic time complexity of Self Attention Blocks (SAB). More details about this architecture can be found in (Lee et al., 2019a). We reproduce the ModelNet 40 classification experiment using 5000 uniformly sampled points for each shape and use a Set Transformer and a Set Sinkformer with two ISAB layers in the encoder and a decoder composed of a SAB and a Pooling by Multihead Attention (PMA) module. While the reported test accuracy is of 87.8% using a Set Transformer, we obtain as our best accuracy when performing 21 iterations of Sinkhorn algorithm within our Sinkformer of 89.1%. Results are summarized in Table 5.1.

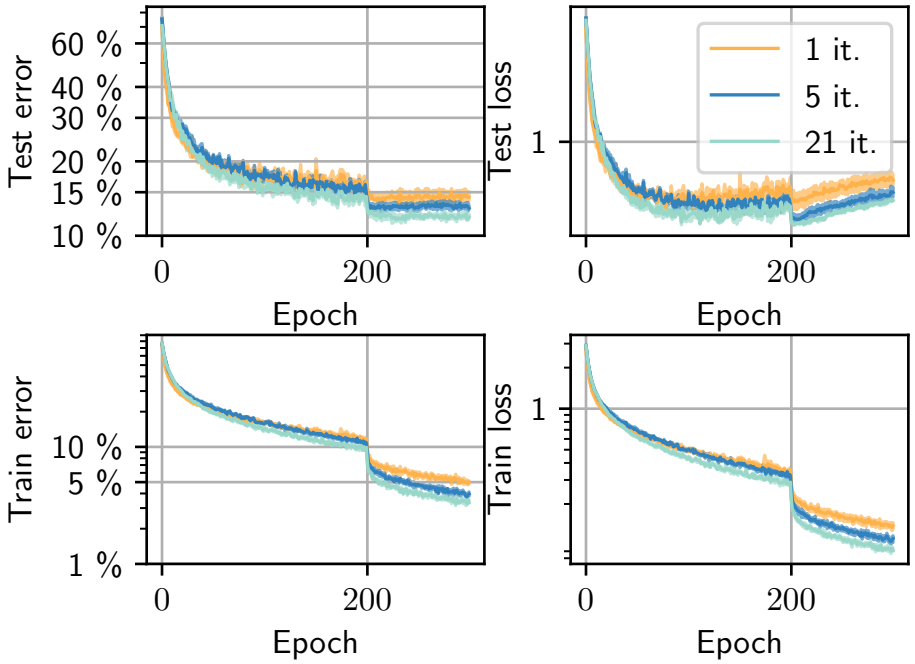

Figure 5.3: **Classification error and loss on ModelNet 40** when training a Set Transformer and a Set Sinkformer with different number of iterations in Sinkhorn's algorithm.

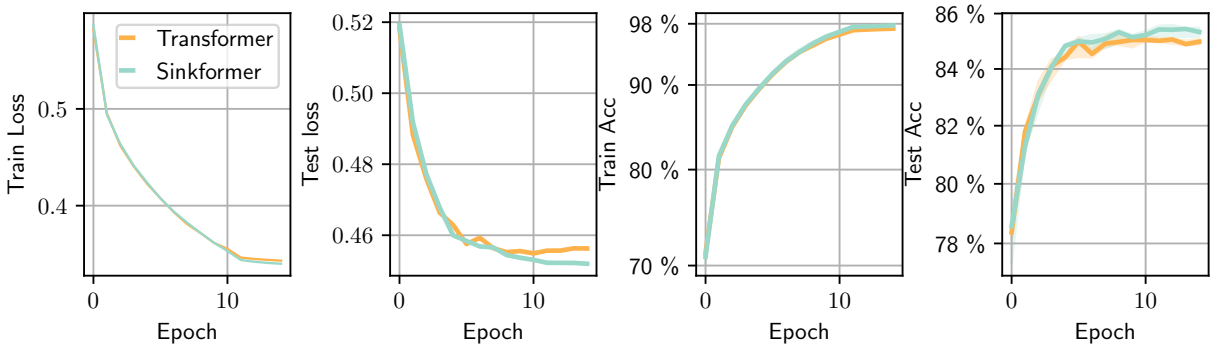

Figure 5.4: **Learning curves** when training a Transformer and a Sinkformer on the Sentiment Analysis task on the IMDb Dataset.

Moreover, we show in Figure 5.3 the learning curves corresponding to this experiment. Interestingly, the number of iterations within Sinkhorn's algorithm increases the accuracy of the model. Note that we only consider an odd number of iterations since we always want to have row-wise stochastic attention matrices to be consistent with the properties of the SoftMax.

Point Cloud Transformers. We also train Point Cloud Transformers (Guo et al., 2021) on ModelNet 40. This architecture achieves accuracy comparable to the state of the art on this dataset. We compare best and median test accuracy over 4 runs. Results are reported in Table 5.1, where we see that while the best test-accuracy is narrowly achieved for the Transformer, the Sinkformer has a slightly better median accuracy.

Table 5.1: **Test accuracy for ModelNet 40** over 4 runs for each model.

Model	Best	Median	Mean	Worst
Set Transformer	87.8%	86.3%	85.8%	84.7%
Set Sinkformer	**89.1**%	**88.4**%	**88.3**%	**88.1**%
Point Cloud Transformer	**93.2**%	92.5%	92.5%	92.3%
Point Cloud Sinkformer	93.1%	**92.8**%	**92.7**%	**92.5**%

5.6.2 Sentiment Analysis

We train a Transformer (composed of an attention-based encoder followed by a max-pooling layer) and a Sinkformer on the IMDb movie review dataset (Maas et al., 2011) for sentiment analysis. This text classification task consists of predicting whether a movie review is positive or negative. The learning curves are shown in Figure 5.4, with a gain in accuracy when using a Sinkformer. In this experiment, Sinkhorn's algorithm converges perfectly in 3 iterations (the resulting attention matrices are doubly stochastic), which corresponds to the green curve. The Sinkformer only adds a small computational overhead, since the training time per epoch is 4m 02s for the Transformer against 4m 22s for the Sinkformer.

5.6.3 Neural Machine Translation

We train a Transformer and its Sinkformer counterpart using the `fairseq` (Ott et al., 2019) sequence modeling toolkit on the IWSLT'14 German to English dataset (Cettolo et al., 2014). The architecture used is composed of an encoder and a decoder, both of depth 6. We plug Sinkhorn's algorithm only into the encoder part. Indeed, in the decoder, we can only pay attention to previous positions in the output sequence. For this reason, we need a mask that prevents a straightforward application of Sinkhorn's algorithm. We demonstrate that even when using the hyper-parameters used to optimally train the Transformer, we achieve a similar BLEU (Papineni et al., 2002) over 6 runs. We first train a Transformer for 30 epochs. On the evaluation set, we obtain a BLEU of 34.43. We then consider a Sinkformer with the weights of the trained Transformer. Interestingly, even this un-adapted Sinkformer provides a median BLEU score of 33.81. We then divide the learning rate by 10 and retrain for 5 additional epochs both the Transformer and the Sinkformer to obtain a median BLEU of respectively 34.68 and 34.73 (Table 5.2). Importantly, the runtime for one training epoch is almost the same for both models: 2m 48s (Transformer) against 2m 52s (Sinkformer).

Table 5.2: **Median BLEU score** over 6 runs on the IWSLT'14 German to English dataset. The score * is when evaluating the Sinkformer with the weights of the trained Transformer.

Model	Epoch 30	Epoch 35
Transformer	34.43	34.68
Sinkformer	$33.81^\star$	**34.73**

5.6.4 Vision Transformers

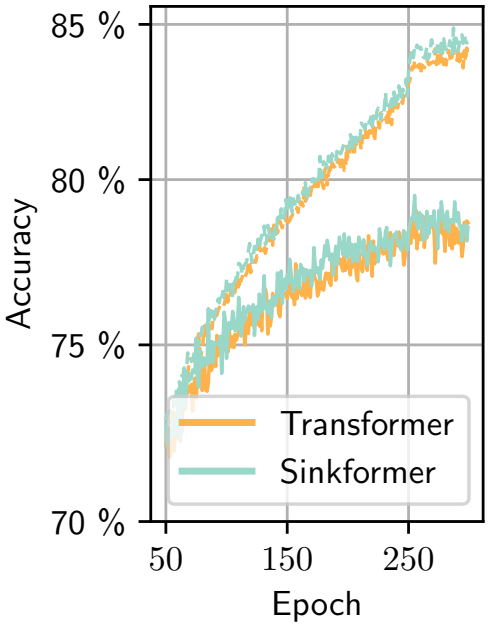

Figure 5.5: **Train** (dotted) and **test** (plain) **accuracy** as a function of the number of epochs when training a ViT and its Sinkformer counterpart on the cats and dogs classification task (median over 5 runs).

Vision Transformers (ViT) (Dosovitskiy et al., 2020) have recently emerged as a promising architecture for achieving state of the art performance on computer vision tasks (Zhai et al., 2021), using only attention based mechanisms by selecting patches of fixed size in images and feeding them into an attention mechanism.

Cats and dogs classification. We train a ViT and its Sinkformer counterpart on a binary cats and dogs image classification task. The evolution of the train and test accuracy is displayed in Figure 5.5. The *median* test accuracy is 79.0% for the Transformer against 79.5% for the Sinkformer, whereas the *maximum* test accuracy is 80.0% for the Transformer against 80.5% for the Sinkformer. We also use 3 iterations in Sinkhorn's algorithm which leads to a negligible computational overhead (training time per epoch of 3m 25s for the Sinkformer against 3m 20s for the Transformer).

Impact of the patch size on the final accuracy. We consider a one-layer and one-head self-attention module on MNIST, with no additional layer. The purpose is to isolate the self-attention module and study how its accuracy is affected by the choice of the patch size. Results are displayed in Figure 5.6. We recall that a MNIST image is of size 28×28. When taking only one patch of size 28, both models are equivalent because the attention matrix is of size 1. However, when the patch size gets smaller, the two models are different and the Sinkformer outperforms the Transformer.

Conclusion

In this chapter, we presented the Sinkformer, a variant of the Transformer in which the SoftMax, which leads to row-wise stochastic attention, is replaced by Sinkhorn's algorithm, which leads to doubly stochastic attention. This new model is motivated by the empirical finding that attention matrices in Transformers get closer and closer to doubly stochastic matrices during the training process. This modification is easily implemented in practice by simply replacing the SoftMax in

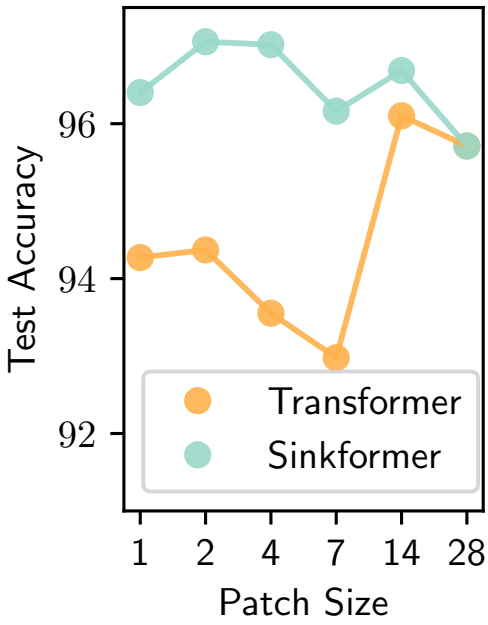

Figure 5.6: **Final test accuracy** when training a one layer and one head self attention module on the MNIST dataset, with no feedforward neural network, when varying the patch size (median over 5 runs).

the attention modules of existing Transformers without changing any parameter in the network. It also provides a new framework for theoretically studying attention-based mechanisms, such as the interpretation of Sinkformers as Wasserstein gradient flows in the infinitesimal step size regime or as diffusion operators in the mean-field limit. On the experimental side, Sinkformers lead to better accuracy in a variety of experiments: classification of 3D shapes, sentiment analysis, neural machine translation, and image classification.

In Section 5.A we give the proofs of all the Propositions and the Theorem. In Section 5.B we present the implementation details of Sinkformers. Section 5.C gives details for the experiments in the chapter.

5.A Proofs

5.A.1 Invariance to the cost function - Proof of Proposition 5.1

Proof. We use the variational formulation for Sinkhorn (Peyré et al., 2019):

$$\texttt{Sinkhorn}(C) = \underset{K\mathbb{1}_n = K^\top \mathbb{1}_n = \mathbb{1}_n}{\operatorname{argmin}} \mathrm{KL}(K|K^0)$$

with

$$\mathrm{KL}(K|K^0) = \sum_{i,j} K_{i,j} \log(\frac{K_{i,j}}{K^0_{i,j}}),$$

where $K^0_{i,j} = \exp(C_{i,j})$.

We let $\tilde{C}_{i,j} = C_{ij} + f_i + g_j$. We have for $K \in U := \{K | K\mathbb{1}_n = K^\top \mathbb{1}_n = \mathbb{1}_n\}$ that $\mathrm{KL}(K|e^{\tilde{C}}) = \sum_{i,j} K_{i,j} \log(\frac{K_{i,j}}{e^{C_{i,j}+f_i+g_j}})$. This gives

$$\mathrm{KL}(K|e^{\tilde{C}}) = \sum_{i,j} K_{i,j}[\log(\frac{K_{i,j}}{e^{C_{i,j}}}) - f_i - g_j] = \sum_{i,j} K_{i,j}[\log(\frac{K_{i,j}}{e^{C_{i,j}}})] - \sum_i f_i - \sum_j g_j$$

so that

$$\mathrm{KL}(K|e^{\tilde{C}}) = \mathrm{KL}(K|e^{C}) - \sum_i f_i - \sum_j g_j.$$

This shows that $\mathrm{KL}(K|e^{\tilde{C}})$ and $\mathrm{KL}(K|e^{C})$ have the same argmin on U which implies that $\texttt{Sinkhorn}(C) = \texttt{Sinkhorn}(\tilde{C})$.

□

5.A.2 PDEs associated with $\mathbf{k^0}, \mathbf{k^1}, \mathbf{k^\infty}$ - Proof of Proposition 5.3

Proof. Recall that for $p \in \{0, 1, \infty\}$, we have $T^p_\mu(x) = \int k^p(x, x')W_V x' d\mu(x')$.

For $h \in \mathcal{C}(\mathbb{R}^d \times \mathbb{R}^d)$ consider

$$\mathcal{H}(\mu) = \int h(x, y)d\mu(x)d\mu(y).$$

Then we have (Santambrogio, 2017)

$$\frac{\delta \mathcal{H}}{\delta \mu}(\mu) = \int (h(x, .) + h(., x))d\mu(x).$$

We can now derive the different gradient expressions for T^0_μ, T^1_μ and T^∞_μ.

For T^0_μ: under Assumption 5.2, we have that $f(x, x') = e^{C(x,x')}$ is symmetric. This gives

$$\frac{\delta \mathcal{F}}{\delta \mu}(\mu) = \int f(., x')d\mu(x')$$

and by differentiation under the integral, under sufficient regularity assumptions on μ, this gives

$$\nabla_W(\mathcal{F}^0)(x) = \int \nabla_x f(x, x')d\mu(x') = \int f(x, x')\nabla_x c(x, x')d\mu(x').$$

Since $\nabla_x c(x, x') = -W_V x'$, we get

$$\nabla_W(\mathcal{F}^0)(x) = -\int e^{c(x,x')}W_V x' d\mu(x').$$

For $\mu = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$ this is exactly

$$\nabla_W(\mathcal{F}^0)(x) = -\sum_{j=1}^n K^0_{i,j} W_V x_j.$$

For T^1_μ: we have

$$\nabla[\log(\int e^{c(.,x')}d\mu(x'))](x) = \int \frac{\nabla_x c(x, x')e^{c(x,x')}}{\int e^{c(x,y)}d\mu(y)}d\mu(x') = -\int \frac{e^{c(x,x')}W_V x'}{\int e^{c(x,y)}d\mu(y)}d\mu(x').$$

For T^2_μ: one has the dual formulation for $\mathcal{F}^\infty$ (Peyré et al., 2019):

$$2\mathcal{F}^\infty(\mu) = -\max_f \int_{\mathbb{R}^d} (f + f^c)\mathrm{d}\mu \tag{5.10}$$

where we denote the soft c transform as

$$f^c(x') := -\log\left(\int e^{f(x)+c(x,x')}\mathrm{d}\mu(x)\right), \tag{5.11}$$

which actually depends on μ and c. One has for an optimal pair $f = f^c$ (Peyré et al., 2019). In addition, one has $k^\infty(x, x') = e^{c(x,x')+f(x)+f(x')}$. The Wasserstein gradient of $\mathcal{F}^\infty$ is then

$$\nabla_W \mathcal{F}^\infty(\mu) = -\nabla f$$

where f is an optimal solution of (5.10) (which is unique up to a constant). The gradient of f can be obtained using (5.11) and the fact that $f = f^c$:

$$\nabla f(x) = -\int e^{f(x)+f(x')+c(x,x')} \nabla_x c(x, x') d\mu(x') = -\int k^\infty(x, x') \nabla_x c(x, x') d\mu(x').$$

This finally gives

$$\nabla_W \mathcal{F}^\infty(\mu) : x \mapsto -\int k^\infty(x, x') W_v x' \mathrm{d}\mu(x'), \tag{5.12}$$

that is what we wanted to show. □

5.A.3 The SoftMax normalization does not correspond to a gradient flow - Proof of Proposition 5.4

Proof. Suppose by contradiction that $T^1_\mu = -\nabla[\log(\int k^0(\cdot, x')d\mu(x'))]$ is a Wasserstein gradient. This implies that there exists a function F such that, $\forall \mu \in \mathcal{M}(\mathbb{R}^d)$ and $\forall x \in \mathbb{R}^d$,

$$\frac{\delta F}{\delta \mu}(\mu)(x) = \log(\int k^0(\cdot, x')d\mu(x')).$$

We therefore have

$$\frac{\delta^2 F}{\delta \mu^2}(\mu)(x, x') = \frac{k^0(x, x')}{\int k^0(x, y)d\mu(y)},$$

$\forall x, x' \in \mathbb{R}^d$. However, $\frac{\delta^2 F}{\delta \mu^2}(\mu)$ is symmetric for all $\mu \in \mathcal{M}(\mathbb{R}^d)$. The relationship $\frac{\delta^2 F}{\delta \mu^2}(\mu)(x, x') = \frac{\delta^2 F}{\delta \mu^2}(\mu)(x', x)$ then implies that for all μ, x and x' such that $k^0(x, x') \neq 0$ we have

$$\int k^0(x, y)d\mu(y) = \int k^0(x', y)d\mu(y).$$

Taking $\mu = \delta_y$ gives $k^0(x, y) = k^0(x', y)$, which by symmetry implies that k^0 is a constant.

This is a contradiction since $k^0(x, x') = \exp(x^\top W_Q^\top W_K x')$. □

5.A.4 Sinkformer's PDE - Proof of Theorem 5.5

Proof. Since $W_Q^\top W_K$ is positive-definite we write it $W_Q^\top W_K = A^2$ where A is positive-definite. Note that thanks to Proposition 5.1, if $\kappa_\varepsilon(x, x') = \exp(-\frac{\|x-x'\|^2}{2\varepsilon})$, one has under Assumption 5.2 that $\kappa^\infty_\varepsilon(Ax, Ax') = k^\infty_\varepsilon(x, x')$. For $x \in \mathbb{R}^d$, we have

$$\overline{T}^\infty_{\mu,\varepsilon}(A^{-1}x) = \frac{1}{\varepsilon}(\int \kappa^\infty_\varepsilon(x, Ax')W_V x' \rho(x')dx' - W_V A^{-1}x).$$

We perform the change of variable $y = Ax'$. This gives

$$\overline{T}^\infty_{\mu,\varepsilon}(A^{-1}x) = \frac{1}{\varepsilon}(\int \kappa^\infty_\varepsilon(x, y)W_V A^{-1} y \rho(A^{-1}y)C_A dy - W_V A^{-1}x),$$

where C_A depends only on A. We then apply Theorem 1 from Marshall and Coifman (2019) with $f = W_V A^{-1}$, $q(x) = \rho(A^{-1}x)$ and $w = \frac{1}{C_A}$, to obtain that

$$\overline{T}^{\infty}_{\mu,\varepsilon}(A^{-1}\cdot) \to \frac{2\nabla f \nabla(q^{1/2})}{q^{1/2}} = W_V A^{-1}\frac{\nabla q}{q}$$

in L^2 norm. Since $q(x) = \rho(A^{-1}x)$ we have obtained that $\frac{\nabla q}{q} = A^{-1}\frac{\nabla \rho}{\rho}(A^{-1}\cdot)$ so that

$$\overline{T}^{\infty}_{\mu,\varepsilon}(A^{-1}x) \to W_V A^{-2}\frac{\nabla \rho}{\rho}(A^{-1}x) = W_V(W_Q^\top W_K)^{-1}\frac{\nabla \rho}{\rho}(A^{-1}x).$$

In other words,

$$\overline{T}^{\infty}_{\mu,\varepsilon} \to W_V(W_Q^\top W_K)^{-1}\frac{\nabla \rho}{\rho},$$

which is exactly what we wanted to show. Note that when $W_V = -W_Q^\top W_K$ this gives the expected result. The general form for the PDE is then

$$\partial_t \rho = \operatorname{div}(-W_V(W_Q^\top W_K)^{-1}\frac{\nabla \rho}{\rho} \times \rho)$$

which gives

$$\partial_t \rho = \Delta \rho$$

if $W_V = -W_Q^\top W_K$. □

5.A.5 Transformer's PDE - Proof of Proposition 5.6

Proof. Let $x \in \mathbb{R}^d$ and consider

$$g_\varepsilon(x) = \varepsilon T^1_{\mu,\varepsilon}(W_Q^{-1}x) = \frac{\int e^{-\frac{\|x - W_K x'\|^2}{2\varepsilon}} W_V x' \rho(x')dx'}{\int e^{-\frac{\|x - W_K x'\|^2}{2\varepsilon}} \rho(x')dx'}.$$

We perform the change of variable $y = W_K x'$. This gives:

$$g_\varepsilon(x) = \frac{\int e^{-\frac{\|x - y\|^2}{2\varepsilon}} W_V W_K^{-1} y \rho(W_K^{-1}y)dy}{\int e^{-\frac{\|x - y'\|^2}{2\varepsilon}} \rho(W_K^{-1}y)dy}.$$

Using the Laplace expansion result from Singer (2006), we obtain that

$$g_\varepsilon(x) = W_V W_K^{-1}\frac{x\rho(W_K^{-1}x) + \frac{\varepsilon}{2}\Delta(x\rho(W_K^{-1}x)) + o(\varepsilon)}{\rho(W_K^{-1}x) + \frac{\varepsilon}{2}\Delta(\rho(W_K^{-1}x)) + o(\varepsilon).}$$

By doing a Taylor expansion for the denominator, we find

$$g_\varepsilon(x) = W_V W_K^{-1}(x + \frac{\varepsilon}{2}\frac{\Delta(x\rho(W_K^{-1}x)}{\rho(W_K^{-1}x)} + o(\varepsilon))(1 - \frac{\varepsilon}{2}\frac{\Delta(\rho(W_K^{-1}x))}{\rho(W_K^{-1}x)} + o(\varepsilon))$$

and

$$g_\varepsilon(x) = W_V W_K^{-1}(x + \varepsilon\frac{\nabla(\rho(W_K^{-1}x))}{\rho(W_K^{-1}x)} + o(\varepsilon).)$$

Since $\overline{T}^1_{\mu,\varepsilon} = T^1_{\mu,\varepsilon} + \frac{1}{\varepsilon}W_Q^\top W_Q = \frac{1}{\varepsilon}(g_\varepsilon(W_Q x) + W_Q^\top W_Q x)$ and because $W_V W_K^{-1} = -W_Q{}^\top W_K W_K^{-1} = -W_Q^\top$ we have

$$\overline{T}^1_{\mu,\varepsilon} = -W_Q^\top W_K^{-1}\frac{\nabla \rho(W_K^{-1}W_Q x))}{\rho(W_K^{-1}W_Q x)} + o(1)$$

which is exactly the expected result. □

5.B Implementation details

We implement Sinkhorn's algorithm in log domain for stability. Given a matrix $K^0 \in \mathbb{R}^{n\times n}$ such that $K^0_{i,j} = e^{C_{i,j}}$ for some $C \in \mathbb{R}^{n\times n}$, Sinkhorn's algorithm (5.3) approaches $(f, g) \in \mathbb{R}^n \times \mathbb{R}^n$ such that $K^\infty = \mathrm{diag}(e^{f^\infty})K^0\mathrm{diag}(e^{g^\infty})$ by iterating in log domain, starting from $g^0 = \not{1}_n$,

$$\begin{aligned} f^{l+1} &= \log(\not{1}_n/n) - \log(Ke^{g^l}) \quad \text{if } l \text{ is even} \\ g^{l+1} &= \log(\not{1}_n/n) - \log(K^\top e^{f^l}) \quad \text{if } l \text{ is odd.} \end{aligned} \tag{5.13}$$

This allows for fast and accurate computations, where $\log(Ke^{g^l})$ and $\log(K^\top e^{f^l})$ are computed using `log-sum-exp`.

5.C Experimental details

5.C.1 ModelNet 40 classification

Set Transformers. For our experiments on ModelNet using Set Transformers, we first prepossess the ModelNet 40 dataset. We then uniformly sample 5000 points from each element in the dataset. Our architecture is composed of two ISAB layers in the encoder and a decoder composed of a SAB and a Pooling by Multihead Attention (PMA) module. For the training, we use a batch-size of 64 and we use Adam (Kingma and Ba, 2014). The training is done over 300 epochs. The initial learning rate is 10^{-3} and is decayed by a factor 10 after 200 epochs.

Point Cloud Transformers. For our experiments on ModelNet using Point Clouds Transformers, we uniformly sample 1024 points from each element in the dataset. For the training, we use a batch-size of 32 and we use SGD (Ruder, 2016). The training is done over 300 epochs. The initial learning rate is 10^{-4} and is decayed by a factor 10 after 250 epochs.

5.C.2 Sentiment Analysis

We use the code available at the repository nlp-turorial[1], where a pretrained Transformer is fine-tuned on the IMDb dataset. In our experiment, we reset the parameters of the pretrained Transformer and train it from scratch on the IMDb dataset. We use an architecture of depth 6, with 8 heads. For the training, we use a batch-size of 32 and we use Adam. The training is done over 15 epochs. The initial learning rate is 10^{-4} and is decayed by a factor 10 after 12 epochs.

5.C.3 Neural Machine Translation

We use the Transformer from `fairseq` and the command for training it on the IWSLT'14[2] dataset. When fine-tuning a Sinkformer, we simply divide the original learning rate by 10.

5.C.4 Vision Transformers

Cats and dogs classification. This experiment is done on the cats and dogs[3] dataset. For this experiment, we use a batch-size of 64 and Adam. We use an architecture of depth 6, with 8 heads, and select a patch-size of 16. The training is done over 300 epochs. The initial learning rate is 5×10^{-5} and divided by 10 after 250 epochs.

[1] https://github.com/lyeoni/nlp-tutorial/tree/master/text-classification-transformer
[2] https://github.com/pytorch/fairseq/blob/main/examples/translation/README.md
[3] https://www.kaggle.com/c/dogs-vs-cats/data

Impact of the patch size on the final accuracy. For this experiment, we use a batch-size of 100 and Adam. We use an architecture of depth 1, with 1 heads, without non-linearity, and select different values for the patch-size. The training is done over 45 epochs. The initial learning rate is 1×10^{-3} (resp. 2×10^{-3}) for the Transformer (resp. Sinkformer) and divided by 10 after 35 epochs and again by 10 after 41 epochs.

Part III

Transformers in Action

6 How do Transformers Perform In-Context Autoregressive Learning?

Transformers have achieved state-of-the-art performance in language modeling tasks. However, the reasons behind their tremendous success are still unclear. In this chapter, towards a better understanding, we train a Transformer model on a simple next token prediction task, where sequences are generated as a first-order autoregressive process $s_{t+1} = Ws_t$. We show how a trained Transformer predicts the next token by first learning W in-context, and then applying a prediction mapping. We call the resulting procedure *in-context autoregressive learning*. More precisely, focusing on commuting orthogonal matrices W, we first show that a trained one-layer linear Transformer implements one step of gradient descent for the minimization of an inner objective function when considering augmented tokens. When the tokens are not augmented, we characterize the global minima of a one-layer diagonal linear multi-head Transformer. Importantly, we exhibit orthogonality between heads and show that positional encoding captures trigonometric relations in the data. On the experimental side, we consider the general case of non-commuting orthogonal matrices and generalize our theoretical findings.

Contents

6.1 Introduction

Transformers (Vaswani et al., 2017) have achieved state-of-the-art performance in natural language processing tasks (Devlin et al., 2018). They now serve as the backbone for large language models, such as GPT (Radford et al., 2018; Brown et al., 2020), Chinchilla (Hoffmann et al., 2022), PaLM (Chowdhery et al., 2023), LLama (Touvron et al., 2023) or Mistral (Jiang et al., 2023). These models, which are causal, are trained to predict the next token s_{T+1} given a sequence (also termed as context) $s_{1:T} \coloneqq (s_1, \cdots, s_T)$. An intriguing property of large Transformers is their ability to adapt their computations given the context $s_{1:T}$. In this work, we make a step towards understanding this *in-context learning ability*. More precisely, assuming the tokens satisfy a relation $s_{T+1} = \varphi_W(s_{1:T})$, with W a context-dependent parameter varying with each sequence, we say that a trained Transformer *autoregressively learns* this relation *in-context* if it decomposes its prediction into 2 steps: first, estimating W through an in-context mapping, and then applying a simple prediction mapping, which is equal or closely related to φ_W (see Definition 6.1).

The goal of this chapter is to fully characterize the autoregressive in-context learning process for optimally-trained Transformers. More precisely, building on the work of Von Oswald et al. (2023b), we focus on a simple autoregressive (AR) process of order 1, where each sequence is generated following the recursion $s_{T+1} = \varphi_W(s_{1:T}) \coloneqq W s_T$, and W is a randomly sampled orthogonal matrix, referred to as the *context matrix*. Such a process is illustrated in dimension 3 in Figure 6.1 for two different matrices W. We investigate the training of a linear Transformer to predict the next token in these AR processes, examining how it estimates W in-context and makes predictions for s_{T+1}. Depending on the input tokens encoding, the in-context mapping can correspond to gradient descent on an inner objective, as suggested by Von Oswald et al. (2023b). Alternatively, the context matrix W might be determined in closed form if the model possesses sufficient expressiveness. This chapter investigates both scenarios.

More precisely, we make the following contributions:

- We begin by reviewing the background and previous works in §6.2. Then, in §6.3, we introduce our autoregressive process, which allows us to mathematically formalize the notion of *in-context autoregressive learning*.
- In §6.4, we demonstrate that if the matrices W commute and the model parameters possess a block structure, then a linear Transformer—trained on augmented tokens as introduced by Von Oswald et al. (2023b)—effectively implements a step of gradient descent on an underlying objective function as in-context mapping.
- In §6.5, we turn our attention to a one-layer linear attention Transformer that incorporates positional encoding but does not use augmented tokens. We comprehensively characterize the minimizers of the training loss. Notably, these minimizers display an orthogonality property across different heads. This aspect underscores the significance of positional encoding in enabling the Transformer to learn geometric operations between tokens through its in-context mapping.

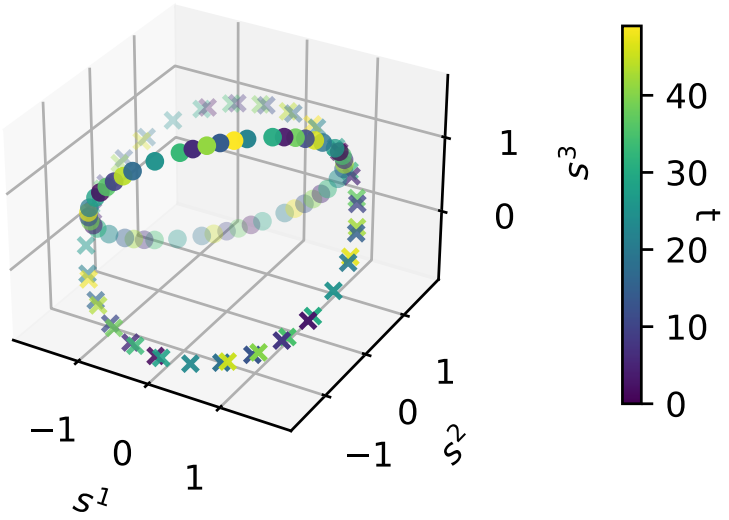

Figure 6.1: **Illustration of the autoregressive process in $\mathbb{R}^3$.** Dots and crosses correspond to two different orthogonal matrices W.

We also study positional-encoding-only attention and show that approximate minimum ℓ_2 norm solutions are favored by the optimization process.

- On the experimental side, in §6.6, we extend our analysis to the more general case where the context matrices W do not commute. We validate our theoretical findings for both augmented and non-augmented scenarios. Furthermore, we explore how variations in the distribution of the context matrices W affect trained positional encodings and lead to structures resembling those of traditional positional encodings commonly used in Transformers.

Each theoretical result of the chapter aims at characterizing the autoregressive in-context learning mechanism for simple models and sequence data. Namely, Propositions 6.5, 6.7 and 6.11 give the structure of the minimizers of the training loss and explicit the corresponding in-context mappings, while Propositions 6.8, 6.9 and 6.12 focus on the optimization process.

Notations. We use lower cases for vectors and upper cases for matrices. $\|.\|$ is the ℓ_2 norm. We denote the transpose and adjoint operators by $^\top$ and *. $O(d)$ (resp $U(d)$) is the orthogonal (resp unitary) manifold, that is $O(d) = \{W \in \mathbb{R}^{d\times d} | W^\top W = I_d\}$ and $U(d) = \{W \in \mathbb{C}^{d\times d} | W^\star W = I_d\}$. The element-wise multiplication is $\odot$. $\langle|\rangle$ is the canonical dot product in $\mathbb{R}^d$, and $\langle\cdot|\cdot\rangle_{\mathbb{C}}$ the canonical hermitian product in $\mathbb{C}^d$. For $\lambda \in \mathbb{C}^d$, λ^k is the element-wise power k of λ: $(\lambda^k)_i = \lambda_i^k$.

6.2 Background and previous works

Causal Language Modelling. Language (or sequence) modeling refers to the development of models to predict the likelihood of observing a sequence $(x_1, \ldots, x_T)$, where each x_t is called a token, and comes from a finite vocabulary. This can be done by using the chain rule of probability $P(X_1 = x_1, X_2 = x_2, \ldots, X_T = x_T) = P(X_1 = x_1) \times P(X_2 = x_2 | X_1 = x_1) \times \ldots \times P(X_T = x_T | X_1 = x_1, \ldots, X_{T-1} = x_{T-1})$ (Jurafsky and Martin, 2009). Predicting these conditional probabilities can be done using a parametrized model $\mathcal{F}_\theta$ to minimize the loss $L(\mathcal{F}_\theta(x_1, ..., x_{T-1}), x_T)$ across all training samples and sequence length T. In common applications, L is chosen as the cross-entropy loss. In other words, the model is trained to predict the next token sequentially. Such a model is called a causal language model: it cannot

access future tokens. Recently, the Transformer has emerged as the model of choice for language modeling.

Transformers. Transformers (Vaswani et al., 2017) process sequences of tokens $(x_1, \dots, x_T)$ of arbitrary length T. In its causal form (Brown et al., 2020; Touvron et al., 2023; Jiang et al., 2023), a Transformer first embeds the tokens to obtain a sequence $(e_1, \dots, e_T)$. It is then composed of a succession of blocks with residual connections (He et al., 2016a). Each block is made of the composition of a multi-head self-attention module and a multi-layer perceptron (MLP). Importantly, the latter acts on each token separately, whereas multi-head self-attention mixes tokens, and corresponds to applying vanilla self-attention in parallel (Michel et al., 2019). More precisely, each multi-head self-attention is parametrized by a collection of weight matrices $(W_Q^h, W_K^h, W_V^h, W_O^h)_{1 \leq h \leq H}$ and returns:

$$(\sum_{h=1}^{H} W_O^h \sum_{t'=1}^{t} \mathcal{A}_{t,t'}^h W_V^h e_{t'})_{t \in \{1, \cdots, T\}}, \tag{6.1}$$

where $\mathcal{A}^h$ is the attention matrix (Bahdanau et al., 2014b) and is usually defined as

$$\mathcal{A}_{t,:}^h = \text{softmax}(\langle W_Q^h e_t, W_K^h e_: \rangle),$$

with $\langle \cdot, \cdot \rangle$ a dot product. The sum over t' in (6.1) stopping at t reflects the causal aspect of the model: the future cannot influence the past. The output at position T is commonly used to predict the next token e_{T+1}. In practice, to help the model encode the relative position of the tokens in the sequence, a positional encoding (PE) is used.

Positional encoding. As described in Kazemnejad et al. (2023), encoding the position in Transformers amounts to defining the dot product $\langle \cdot, \cdot \rangle$ in the attention matrix, using additional (learnable or not) parameters. Popular designs include Absolute PE (Vaswani et al., 2017), Relative PE (Raffel et al., 2020), AliBI (Press et al., 2021), Rotary (Su et al., 2024), and NoPE (Kazemnejad et al., 2023). In this chapter, we consider a learnable positional encoding.

Linear attention. In its simplest form, linear attention (Katharopoulos et al., 2020) consists in replacing the softmax in (6.1) by the identity. More formally, it consists in considering that each coefficient in the attention matrix is $\mathcal{A}_{t,t'}^h = \langle W_K^h e_{t'}, W_Q^h e_t \rangle$. The main practical motivation of linear attention is that it enables faster inference (Katharopoulos et al., 2020; Fournier et al., 2023). Note that even though they are called linear Transformers, the resulting models are non-linear with respect to the input sequence and jointly non-linear with respect to the parameters. From a theoretical perspective, linear attention has become the model of choice to understand the in-context-learning properties of Transformers (Mahankali et al., 2023; Ahn et al., 2023; Zhang et al., 2023).

In-context-Learning in Transformers The seminal work of Brown et al. (2020) reported an in-context-learning phenomenon in Transformer language models: these models can solve few-shot learning problems given examples in-context. Namely, given a sequence $(x_1, f(x_1), x_2, f(x_2), \cdots, x_n)$, a trained Transformer can infer the next output $f(x_n)$ without additional parameter updates. This surprising ability has been the focus of recent research. Some works consider the softmax attention without considering training dynamics (Garg et al., 2022; Akyürek et al., 2022; Li et al., 2023). Other works focus solely on linear attention and characterize the minimizers of the training loss when f is sampled across linear forms on $\mathbb{R}^d$, that is $f(x) = w^\top x$ for some

w (Mahankali et al., 2023; Ahn et al., 2023; Zhang et al., 2023). In particular, these works discuss the ability of Transformers to implement optimization algorithms in their forward pass at inference, as empirically suggested by Von Oswald et al. (2023a). Nevertheless, the formulations used by Von Oswald et al. (2023a); Mahankali et al. (2023); Ahn et al. (2023); Zhang et al. (2023) are all based on concatenating the tokens so that the Transformer's input takes the form $\begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ f(x_1) & f(x_2) & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{(d+1)\times n}$. However, the necessity for this concatenation limits the impact of these results as there is no guarantee that the Transformer would implement this operation in its first layer. In addition, these works explicitly consider the minimization of an in-context loss, which is different from the next-token prediction loss in causal Transformers. In contrast, our work considers the next-token prediction loss and considers a more general notion of in-context learning, namely *in-context autoregressive learning*, that we describe in the next section.

6.3 Linear Attention for AR Processes

Token encoding. Building on the framework established by Von Oswald et al. (2023b), we consider a noiseless setting where each sequence begins with an initial token $s_1 = 1_d$. This token acts as a start-of-sentence marker. The subsequent states are generated according to $s_{t+1} = Ws_t$, where W is a matrix referred to as the *context matrix*. This matrix is sampled uniformly from a subset $\mathcal{C}_O$ (respectively, $\mathcal{C}_U$) of $O(d)$ (respectively, $U(d)$), and we denote $\mathcal{W}$ as the corresponding distribution: $W \sim \mathcal{W} := \mathcal{U}(\mathcal{C})$. Considering norm-preserving matrices ensures the stability of the AR process, which is crucial to be able to learn from long sequences (i.e. using large T). In this chapter, we contrast $\mathcal{C}_O$ and $\mathcal{C}_U$ to showcase how the distribution of in-context parameters W impacts the in-context mapping learned by Transformers. In addition, we have the following.

Remark 1. *If $W_U \in U(d)$, then*

$$W_O := \begin{bmatrix} \mathrm{Real}(W_U) & -\mathrm{Imag}(W_U) \\ \mathrm{Imag}(W_U) & \mathrm{Real}(W_U) \end{bmatrix}$$

is in $O(2d)$ and has pairwise conjugate eigenvalues. W_O is a rotation (because W_O is similar to a 2×2 block diagonal matrix with rotations). Reciprocally, for any rotation W_O of size $2d$ corresponds a unitary matrix W_U of size d by selecting half of the eigenvalues (for instance those with positive imaginary parts).

Therefore, $U(d)$ can be viewed as a subset of $O(2d)$, while $O(d) \subset U(d)$. As such, placing ourselves in $U(d)$ corresponds to a compact way of considering real AR processes in dimension $2d$.

In our analysis, we consider two settings in which the sequence $s_{1:T}$ is mapped to a new sequence $e_{1:T}$. In the *augmented setting* (§6.4), the tokens are defined as $e_t := (0, s_t, s_{t-1})$, aligning with the setup used by Von Oswald et al. (2023b). In contrast, the *non-augmented setting* (§6.5) utilizes a simpler definition where the tokens are simply $e_t := s_t$.

Model and training process. We consider a Transformer with linear attention, which includes an optionally trainable positional encoding $P \in \mathbb{R}^{T_{\max} \times T_{\max}}$ for some $T_{\max} \in \mathbb{N}$:

$$\mathcal{A}^h_{t,t'} = P_{t,t'} \langle W^h_Q e_t | W^h_K e_{t'} \rangle. \tag{6.2}$$

Throughout this chapter, we will re-parameterize the model by setting $B^h = W^h_O W^h_V$ and $A^h = W^{h\top}_K W^h_Q$. Note that such an assumption is standard in theoretical studies on the training

of Transformers (Mahankali et al., 2023; Zhang et al., 2023; Ahn et al., 2023). The trainable parameters are therefore $\theta = ((A^h, B^h, P))_{1\leq h\leq H}$ when the positional encoding is trainable and $\theta = ((A^h, B^h))_{1\leq h\leq H}$ otherwise. This defines a mapping $\mathcal{T}_\theta(e_{1:T})$ by selecting a section from some element τ in the output sequence (6.1). We focus on the population loss, defined as:

$$\ell(\theta) \coloneqq \sum_{T=2}^{T_{\max}} \mathbb{E}_{W\sim\mathcal{W}} \|\mathcal{T}_\theta(e_{1:T}) - s_{T+1}\|^2, \tag{6.3}$$

indicating the model's objective to predict s_{T+1} given $e_{1:T}$. It is important to note that both s_{T+1} and $e_{1:T}$ appearing in (6.3) are computed from a random W and are therefore random variables.

In-context autoregressive learning. Our goal is to theoretically characterize the parameters θ^* that minimize ℓ, discuss the convergence of gradient descent to these minima, and characterize the in-context autoregressive learning of the model. This learning process is defined as the model's ability to learn and adapt within the given context: first by estimating W (or more generally some power of W) using an in-context mapping Γ, then by predicting the next token using a simple mapping ψ. In the context or AR processes, we formalize this procedure in the following definition.

Definition 6.1 (In-context autoregressive learning)**.** *We say that $\mathcal{T}_{\theta^*}$* learns autoregressively in-context *the AR process $s_{T+1} = Ws_T$ if $\mathcal{T}_{\theta^*}(e_{1:T})$ can be decomposed in two steps: (1) first applying an in-context mapping $\gamma = \Gamma_{\theta^*}(e_{1:T})$, (2) then using a prediction mapping $\mathcal{T}_{\theta^*}(e_{1:T}) = \psi_\gamma(e_{1:T})$. This prediction mapping should be of the form $\psi_\gamma(e_{1:T}) = \gamma s_\tau$ for some shift $\tau \in \{1, \cdots, T\}$. With such a factorization, in-context learning arises when the training loss $\ell(\theta^*)$ is small. This corresponds to having $\Gamma_{\theta^*}(e_{1:T}) \approx W^{T+1-\tau}$ when applied to data $e_{1:T}$ exactly generated by the AR process with matrix W.*

In this work, we will have either $\tau = T$ or $\tau = T - 1$.

Remark 2. *We use the word in-context to make explicit the fact that the matrix W is different for each sequence. As a consequence, attention-based models are particularly well suited to such a task because predicting W involves considering relationships between tokens. In contrast, RNNs perform poorly in this setting precisely because they do not consider interactions between tokens. In fact, in its simplest form, a linear RNN with parameters A and B outputs, for each t: $y_t = \sum_{k=1}^{t} A^{t-k}BW^{k-1}e_1$. It is easy to see that A and B would have to depend on W for y_t to be close to $W^t s_1$, which is impossible because W is different for each sequence.*

To fully characterize the in-context mapping Γ and prediction mapping ψ, we rely on a commutativity assumption.

Assumption 6.2 (Commutativity)**.** *The matrices W in $\mathcal{C}$ commute. Hence, they are co-diagonalizable in a unitary basis of $\mathbb{C}^{d\times d}$. Up to a change of basis, we therefore suppose $\mathcal{C}_U = \{\mathrm{diag}(\lambda_1, \cdots, \lambda_d), |\lambda_i| = 1\}$ and $\mathcal{C}_O = \{(\lambda_1, \bar{\lambda_1}, \cdots, \lambda_\delta, \bar{\lambda_\delta}), |\lambda_i| = 1\}$, with $d = 2\delta$.*

For conciseness, we only consider pairwise conjugate eigenvalues in $\mathcal{C}_O$. While assumption 6.2 is a strong one, it is a standard practice in the study of matrix-involved learning problems (Arora et al., 2019). We highlight that, to the best of our knowledge, this is the first work that provides a theoretical characterization of the minima of ℓ. The general problem involving non-commutative matrices is complex, and we leave it for future work. Note that recent studies, such as those by Mahankali et al. (2023); Zhang et al. (2023); Ahn et al. (2023) focus on linear regression problems $x \mapsto w^\top x$, which rewrites $x \mapsto 1_d^\top \mathrm{diag}(w)x$. Therefore, considering diagonal matrices

is a natural extension of these approaches to autoregressive settings. Note also that imposing commutativity, while a simplification, represents a practical method of narrowing down the class of models φ_W. Indeed, in high dimension, it becomes necessary to restrict the set $\mathcal{C}$, otherwise, W cannot be accurately estimated when $T < d$. Note that however, in §6.6, we experimentally consider the general case of non-commuting matrices.

6.4 In-context mapping with gradient descent

In this section, we consider the augmented tokens $e_t \coloneqq (0, s_t, s_{t-1})$. We show that under assumption 6.2 and an additional assumption on the structure of θ at initialization, the minimization of (6.3) leads to the linear Transformer implementing one-step of gradient descent on an inner objective as its in-context mapping Γ_{θ^*}. The motivation behind this augmented dataset is that the tokens e_t can be computed after a two-head self-attention layer with a softmax. Indeed, we have the following result.

Lemma 6.3. *The tokens $e_{1:T}$ can be approximated with arbitrary precision given tokens $s_{1:T}$ with a Transformer (6.1).*

For a proof, see Appendix 6.A.1. We suppose that $\mathcal{W} = \mathcal{U}(\mathcal{C}_U)$, that is we consider unitary matrices. We consider $\mathcal{T}_\theta$ to be a one-head attention layer with skip connection and output the fist d coordinates of the token T. More precisely, one has

$$\mathcal{T}_\theta(e_{1:T}) = \left(e_T + \sum_{t=1}^{T} \langle Ae_T | e_t \rangle_{\mathbb{C}} B e_t \right)_{1:d}. \tag{6.4}$$

Importantly, we do not consider a positional encoding as the relative position is already stored in each token e_t. Note that we use the hermitian product $\langle | \rangle_{\mathbb{C}}$ as $e_t \in \mathbb{C}^{3d}$.

We have the following result showing the existence of θ_0 such that (6.4) corresponds to one step of gradient descent on an inner objective.

Proposition 6.4 (Von Oswald et al. (2023b))**.** *There exists θ_0 such that $\Gamma_{\theta_0}(e_{1:T}) = W_0 - \eta \nabla L(W_0, e_{1:T})$ with*

$$L(W, e_{1:T}) = \frac{1}{2} \sum_{t=1}^{T-1} \|s_{t+1} - W s_t\|^2, \tag{6.5}$$

and W_0 is any gradient descent initialization.

We now make the following assumption on the structure of A and B.

Assumption 1. *We parameterize A and B as*

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & A_1 & A_2 \\ 0 & A_3 & A_4 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 0 & B_1 & B_2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

with $A_i = a_i I$ and $B_i = b_i I$.

Importantly, while the zero block structure is stable with gradient descent on loss (6.3), we do impose the non-zero blocks to stay diagonal during training. Note that considering diagonal matrices is a widely used assumption in the topic of linear diagonal networks (Woodworth et al., 2020; Pesme et al., 2021). Note also that we consider the general parametrization for A and B in our experiments in §6.6.

Under assumption 1, we have the following result, stating that at optimality, Γ_{θ^*} corresponds to Γ_{θ_0} in Proposition 6.5 with $W_0 = 0$.

Proposition 6.5 (In-context autoregressive learning with gradient-descent)**.** *Suppose $\mathcal{C} = \mathcal{C}_U$, assumptions 6.2 and 1. Then loss (6.3) is minimal for θ^* such that $a_1^* + a_4^* = a_2^* = b_2^* = 0$ and $a_3^* b_1^* = \frac{\sum_{T=2}^{T_{\max}} T}{\sum_{T=2}^{T_{\max}} (T^2+(d-1)T)}$. Furthermore, an optimal in-context mapping Γ_{θ^*} is one step of gradient descent starting from the initialization $\lambda = 0$, with a step size asymptotically equivalent to $\frac{3}{2T_{\max}}$ with respect to $T_{\max}$.*

For a full proof, see Appendix 6.A.3. Proposition 6.5 demonstrates that a single step of gradient descent constitutes the optimal forward rule for the Transformer $\mathcal{T}_\theta$. This finding aligns with recent research showing that one step of gradient descent is the optimal in-context learner for one layer of self-attention in the context of linear regression (Mahankali et al., 2023; Zhang et al., 2023; Ahn et al., 2023). However, a substantial drawback of considering augmented tokens is that it requires previous layers to form these tokens which—although is possible according to Lemma 6.3—is a strong assumption. Therefore, in the next section, we consider the non-augmented setting where we do not make strong assumptions about previous layers.

6.5 In-context mapping as a geometric relation

In this section, we consider the non-augmented tokens where $e_t := s_t$, and a multi-head self-attention model $\mathcal{T}_\theta$:

$$\mathcal{T}_\theta(e_{1:T}) = \sum_{h=1}^{H} \sum_{t=1}^{T} P_{T-1,t} \langle e_t | A^h e_{T-1} \rangle_{\mathbb{C}} B^h e_t, \tag{6.6}$$

that is we consider $\tau = T-1$, the second to last token in the output, and no residual connections. While not considering the last token in the output is not done in practice, this small modification is necessary to achieve zero population loss. We stress out that *we still mask* the token we want to predict.

We consider a self-attention module with H heads, and we define the following assumption.

Assumption 2. *A^h and B^h are diagonal for all h: $A^h = \mathrm{diag}(a_h)$ and $B^h = \mathrm{diag}(b_h)$ with $(a_h, b_h) \in \mathbb{R}^d \times \mathbb{R}^d$.*

Importantly, we impose the diagonal structure during training. This diagonal aspect reflects the diagonal property of the context matrices. Under assumptions 6.2 and 2, we have the following result.

Lemma 6.6. *Suppose assumptions 6.2 and 2. Writing $a_h = (a_h^1, \cdots, a_h^d)$ and $b_h = (b_h^1, \cdots, b_h^d)$ and letting $\mathtt{A} := (a^1, \cdots, a^d) \in \mathbb{R}^{H \times d}$, and $\mathtt{B} := (b^1, \cdots, b^d)$, one has for an input sequence $e_{1:T} = (1_d, \lambda, \cdots, \lambda^{T-1}) \in \mathbb{R}^{T \times d}$:*

$$\mathcal{T}_\theta(e_{1:T}) = \sum_{t=1}^{T} P_{T-1,t} [\mathtt{B}^\top \mathtt{A}] \lambda^{t-T+1} \odot \lambda^{t-1}.$$

For a full proof, refer to appendix 6.A.4. Note that $\mathtt{A}$ and $\mathtt{B}$ correspond to the concatenation of diagonals $W_K^{h\top} W_Q^h$ and $W_O^h W_V^h$ along heads.

It is easily seen from Lemma 6.6 that the choice of $P^*_{T-1,t} = \delta_{t=T}$ and ${\mathtt{B}^*}^\top \mathtt{A}^* = I_d$ implies $\mathcal{T}_{\theta^*}(e_{1:T}) = \lambda^T$, and therefore $\ell(\theta^*) = 0$. We see that this requires at least d heads. A natural

question is whether there are other optimal solutions and how to characterize them. To answer this question, we investigate the case of unitary context matrices before moving to orthogonal ones. We consider these cases separately because they both lead to different in-context mappings. We recall that $U(d)$ can be seen as a subset of $O(2d)$ (see Remark 1).

6.5.1 Unitary context matrices.

In this case, coefficients in the context matrices are drawn independently. This constrains the possible values for θ^* achieving zero loss. Indeed, we have the following result.

Proposition 6.7 (Unitary optimal in-context mapping)**.** *Suppose assumptions 6.2 and 2. Any $\theta^* = (\mathtt{A}^*, \mathtt{B}^*, P^*)$ achieving zero of the loss (6.3) satisfies $P^*_{T-1,t} = 0$ if $t \neq T$, $P^*_{T-1,T}({\mathtt{B}^*}^\top \mathtt{A}^*)_{ii} = 1$, and $({\mathtt{B}^*}^\top \mathtt{A}^*)_{ij} = 0$ for $i \neq j$. Therefore, one must have $H \geq d$. An optimal in-context mapping satisfies $\Gamma_{\theta^*}(e_{1:T}) = \bar{e}_{T-1} \odot e_T$ and the predictive mapping $\psi_\gamma(e_{1:T}) = \gamma \odot e_T$.*

Proof sketch. At $\ell(\theta^*) = 0$ one has $\mathcal{T}_{\theta^*}(e_{1:T}) = \lambda^T$. We notice that $\mathcal{T}_{\theta^*}(e_{1:T})$ is a polynomial in the λ_i's. Identifying the coefficients leads to the desired results. □

For a full proof, refer to appendix 6.A.5. In particular, for $e_{1:T} = (1_d, \lambda, \cdots, \lambda^{T-1})$, we have $\Gamma_{\theta^*}(e_{1:T}) = \lambda$, and $\psi_{\Gamma_{\theta^*}}(e_{1:T}) = \lambda \odot \lambda^{T-1} = \lambda^T$.

Orthogonality. The equality $(\mathtt{B}^\top \mathtt{A})_{ij} = 0$ for $i \neq j$ corresponds to an orthogonality property between heads. Indeed, to further understand what Proposition 6.7 implies in terms of learned model, let's look at the particular case in which $H = d$ and, at optimality, $\mathtt{A}^* = \mathtt{B}^* = I_d$ and $P^*_{T-1,t} = \delta_{t=T}$. Therefore, the positional encoding selects the last token in the input sequence, hence learning the structure of the training data. In parallel, each attention matrix captures a coefficient in λ: $(\mathcal{T}_{\theta^*}(e_{1:T}))_h = \lambda_h \lambda_h^{T-1}$. When there are more than d heads, some heads are useless, and can therefore be pruned. Such a finding can be related to the work of Michel et al. (2019), where the authors experimentally show that some heads can be pruned without significantly affecting the performance of Transformers. Orthogonality in the context of Transformers was also investigated by directly imposing orthogonality between the outputs of each attention head (Lee et al., 2019b) or on attention maps (Chen et al., 2022; Zhang et al., 2021). The ability of the positional encoding to recover the spatial structure was already shown by Jelassi et al. (2022), which studies Vision Transformers (Dosovitskiy et al., 2020).

Convergence of gradient descent. Now that we have characterized all the global minima of the loss (6.3), we can study the convergence of the optimization process. We have the following Proposition, which shows that the population loss (6.3) writes as a quadratic form in $\mathtt{B}^\top \mathtt{A}$ and P, which enables connections with matrix factorization.

Proposition 6.8 (Quadratic loss)**.** *Under assumptions 6.2 and 2, loss (6.3) reads*

$$\ell(\mathtt{A}, \mathtt{B}, P) = \sum_{T=2}^{T_{\max}} l(\mathtt{B}^\top \mathtt{A}, P_{T-1})$$

with $l(\mathtt{C}, p) = \|p\|_2^2 \|\mathtt{C}\|_F^2 + p_{T-1}^2 S(\mathtt{C}^\top \mathtt{C}) - 2\mathrm{Tr}(\mathtt{C}) p_T + d$, where S is the sum of all coefficients operator.

A proof is in Appendix 6.A.6. For an optimal P^* with $P^*_{T-1,t} = \delta_{t=T}$, $\ell(\mathtt{A}, \mathtt{B}, P^*) = (T_{\max} - 1)\|\mathtt{B}^\top \mathtt{A} - I\|_F^2$, for which we can use Theorem 2.2 of Nguegnang et al. (2021) to argue that for almost all initial values, gradient flow on ℓ will converge to a global minimum, that is $\mathtt{B}^\top \mathtt{A} = I_d$.

When training is also done on $p_T \coloneqq P_{T-1,T}$, the loss is then $\ell(\mathtt{A}, \mathtt{B}, P) = \sum_{T=2}^{T_{\max}} \|p_T \mathtt{B}^\top \mathtt{A} - I\|_F^2$. Note that even for $T_{\max} = 2$, convergence of gradient descent in $(\mathtt{A}, \mathtt{B}, p_2)$ on ℓ to a global minimum is an open problem, for which a conjecture (Nguegnang et al., 2021; Achour et al., 2021) states that for almost all initialization, $(\mathtt{A}, \mathtt{B}, p_2)$ will converge to a global minimum of ℓ. We provide evidence for global convergence in Figure 6.3. Yet, we have the following result in the scalar case $H = d = 1$. Its proof is in Appendix 6.A.7.

Proposition 6.9. *Consider the loss $\ell(a, b, p) = (pab - 1)^2$. Suppose that at initialization, $|pab - 1| < 1$. Then gradient flow on (a, b, p) converges to a global minimum satisfying $a^* b^* p^* = 1$.*

Role of the softmax**.** Our results rely heavily on the use of linear attention. In fact, we could not find a natural way to express the global minimum of the training loss (6.3) when a softmax layer was involved, even in dimension $d = 1$. To gain more insight, we conducted an experiment where we trained different models with and without softmax and MLP layers. The results are shown in Figure 6.9 in Appendix 6.B, where it is clear that in the case of commuting context matrices, using a softmax is incompatible with learning the underlying in-context mapping.

6.5.2 Orthogonal context matrices.

We now turn to the case where the context matrices are in $\mathcal{C}_O$. We recall that this imposes that the λ_i are pairwise conjugate. Therefore, the dimension d is even, and we write it $d = 2\delta$. The context matrices are therefore rotations. This property changes the optimization landscape and other solutions are possible, as shown in the following Lemma.

Lemma 6.10. *Suppose assumptions 6.2, 2. If $P^*_{T-1,T-1} = -1$, $P^*_{T-1,T} = 2$ and 0 otherwise, and $\mathtt{B}^{*\top}\mathtt{A}^* = \frac{1}{2}\operatorname{diag}(J, \cdots, J)$, with $J \in \mathbb{R}^{2\times 2}$ and $J_{ij} = 1$ for all i, j, then $\mathcal{T}_{\theta^*}(e_{1:T}) = \lambda^T$.*

For a full proof, refer to Appendix 6.A.5. In this case, $H = \delta$ heads are sufficient to reach zero population loss. The optimal parameters can be exactly characterized.

Proposition 6.11 (Orthogonal optimal in-context mapping)**.** *Suppose assumptions 6.2 and 2. Any $\theta^* = (\mathtt{A}^*, \mathtt{B}^*, P^*)$ with $\ell(\theta^*) = 0$ in (6.3) satisfies, denoting $\mathtt{C}^* = \mathtt{B}^{*\top}\mathtt{A}^*$ and $p^* = P^*_{T-1}$: $p^*_t = 0$ if $t < T - 1$, $p^*_T \mathtt{C}^*_{i,i} = 1$, $p^*_T \mathtt{C}^*_{2i-1,2i} + (\mathtt{C}^*_{2i-1,2i-1} + \mathtt{C}^*_{2i-1,2i}) p^*_{T-1} = 0$, $p^*_T \mathtt{C}^*_{2i,2i-1} + (\mathtt{C}^*_{2i,2i} + \mathtt{C}^*_{2i,2i-1}) p^*_{T-1} = 0$, $\mathtt{C}^*_{2i-1,j} = \mathtt{C}^*_{2i,j} = 0$ for $j \neq 2i - 1, 2i$. An optimal in-context mapping is then, for $e_t = \lambda^{t-1}$: $\Gamma_{\theta^*}(e_{1:T}) = \lambda^2$, and the corresponding predictive mapping $\psi_{\Gamma_{\theta^*}(e_{1:T})}(e_{1:T}) = \lambda^2 \odot e_{T-1} = \lambda^T$.*

Proof sketch. Similarly to Proposition 6.7, we identify the coefficients of a polynomial, with careful inspection of terms involving pairwise conjugate contexts $(\lambda_i, \bar{\lambda_i})$. □

Similarly to Proposition 6.7, this result indicates an orthogonal property between heads. A closer look at the computation of Γ_{θ^*} reveals that the relation implemented in-context by the Transformer in Proposition 6.11 is an extension of a known formula in trigonometry: $2\cos\theta R_\theta - I_2 = R_{2\theta}$, with R_θ the rotation of parameter θ in $\mathbb{R}^2$ (see Figure 6.2). Importantly, when $\delta \leq H < 2\delta = d$, the optimal $\mathtt{C}^*$ in Proposition 6.11 is of rank δ, which corresponds to Lemma 6.10. However, when $H \geq d$, full-rank solutions are achievable.

Under the assumptions of Proposition 6.7, the population loss (6.3) is also a quadratic form in P and $\mathtt{B}^\top\mathtt{A}$. Similarly to Proposition 6.8, global convergence results of gradient descent on such loss function are still an open problem (Achour et al., 2021). We provide experimental evidence for convergence in Figure 6.3.

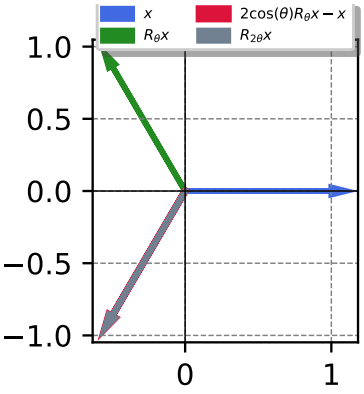

Figure 6.2: **Trigonometric formula** implemented by the Transformer in-context. The minima of the training loss correspond to implementing, up to multiplying factors: $2\cos\theta R_\theta - I_2 = R_{2\theta}$.

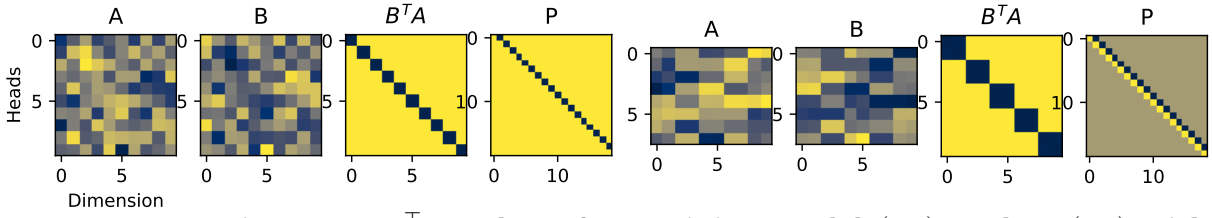

Figure 6.3: **Matrices $\mathtt{A}$, $\mathtt{B}$, $\mathtt{B}^\top\mathtt{A}$ and P after training model (6.6) on loss (6.3) with random initialization**. We take $d = 10$ and $T = 15$. **Left:** Unitary context case with $H = 10$. **Right:** Orthogonal context case, with $H = 8 < d$, which leads to low rank $\mathtt{B}^\top\mathtt{A}$. In both cases, we obtain arbitrarily small final loss. We recover parameters corresponding to our Propositions 6.7 and 6.11.

6.5.3 Positional encoding-only attention.

We end this section by investigating the impact of the context distribution on the trained positional encoding P. For this, we consider a positional encoding-only Transformer, that is, we fix $\mathtt{B}^\top\mathtt{A} = I_d$. In this case, the problem decomposes component-wise, and we only need to consider the $d = 1$ case. We therefore consider the AR process $s_{t+1} = \lambda s_t$ for $|\lambda| = 1$. We break the symmetry of the context distribution: for $\mu \geq 1$ and $\theta \sim \mathcal{U}(0, 2\pi)$, we define $\lambda = e^{i\theta/\mu}$. We denote $\mathcal{W}(\mu)$ as the corresponding distribution. Therefore, we focus on the optimization problem:

$$\min_{p\in\mathbb{R}^T} l(p) := \mathbb{E}_{\lambda\sim\mathcal{W}(\mu)} |\sum_{t=1}^{T} p_t \lambda^{2t-T} - \lambda^T|^2. \tag{6.7}$$

Here again, the same proof as for Proposition 6.7 shows that the optimal positional encoding is $p^* = \delta_{t=T}$, meaning that we predict the next token using the last token in the context. However, depending on μ, (6.7) can be ill-conditioned.

Proposition 6.12 (Conditioning). *The Hessian $H \in \mathbb{R}^{T\times T}$ of l in (6.7) is given by*

$$H_{t,t'} = \frac{\mu}{4\pi(t'-t)} \sin(4(t'-t)\frac{\pi}{\mu}).$$

Denoting $\sigma_1(\mu) \geq \cdots \geq \sigma_T(\mu)$ its eigenvalues, one has $\sigma_1(\mu) \to T$ and $\sigma_{t>1}(\mu) \to 0$ as $\mu \to +\infty$.

Therefore, for large μ, H in Proposition 6.7 is poorly conditioned. In such a setting, gradient descent, even with a large number of iterations, induces a ℓ_2 regularization (Yao et al., 2007). As an informal consequence, approximate solutions computed by gradient descent significantly deviate from the optimal p^*. As demonstrated experimentally in Figure 6.6 and §6.6, the effect of this regularization is a spatial smoothing of the positional encoding, which leads to entirely

different in-context mappings Γ, hence showing the effect of the optimization process on the in-context autoregressive learning abilities of Transformers.

6.6 Experiments

In this section, we illustrate and extend our results through experiments. Our code in Pytorch (Paszke et al., 2017) and JAX (Bradbury et al., 2018) is open-sourced at `https://github.com/michaelsdr/ical`. We use the standard parametrization of Transformers, that is we train on (W_Q, W_K, W_V, W_O).

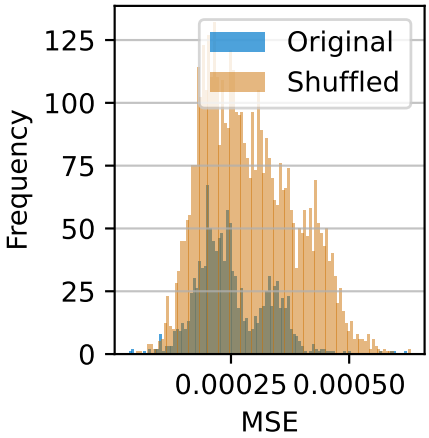

Figure 6.4: **Histograms** of the mean squared errors (MSE) when fitting an AR process to sequences in D (original, in blue) or D_{shuffle} (shuffled, in orange). We only display MSEs bigger than a threshold of 10^{-12}.

Validation of the token encoding choice. Throughout the chapter, we assume that the $s_{1:T}$ are generated following an AR process $s_{t+1} = Ws_t$. Even though we acknowledge that the AR process is an overly simplistic model for real-word sentences, we provide empirical justification for using it by showing that such a process better explains real data than random ones. We use the `nltk` package (Bird et al., 2009), and we employ classic literary works, specifically 'Moby Dick' by Herman Melville sourced from Project Gutenberg. We use the tokenizer and word embedding layer of a pre-trained GPT-2 model (Radford et al., 2019), and end up with about 325000 token representations in dimension $d = 1280$, that we reformat in a dataset D of shape (n, T, d), where $T = 5$ (we keep the relative order of each token). We also consider a shuffled counterpart of D where the shuffling is done across the first two dimensions. In other words, we create a dataset D_{shuffle} from a permutation of the tokens of the book.
We then fit AR processes for each sequence in the two datasets using loss (6.5), which we minimize by solving a linear system. It should be noted that the problem remains non-trivial for some sequences, despite T being significantly smaller than d. This complexity arises because certain sequences might contain identical elements with differing successors or predecessors. We hypothesize that when sequences are shuffled, the number of such inconsistencies increases since the language's structure is lost. This hypothesis is validated in Figure 6.4, where we display the histograms of the fitting losses when they are bigger than 10^{-12}. There are 4 times more sequences with such an error for the shuffled dataset than for the original. This shows that the AR process is better suited when data present some semantics.

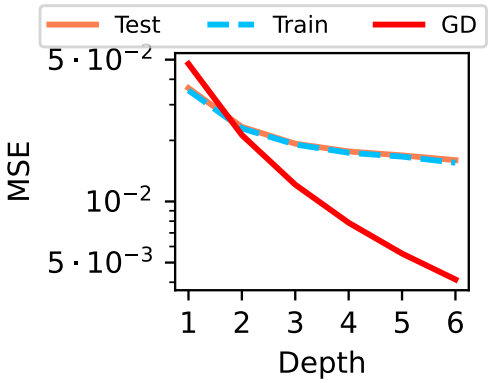

Figure 6.5: **Evolution** of the MSE with depth L. We compare with L steps of gradient descent on the inner loss (6.5). At initialization, the MSE is between 1 and 2.

Augmented setting. We investigate whether the results of §6.4 still hold without assumptions 6.2 and 1. We consider the model (6.4) on the augmented tokens $e_t = (0, s_t, s_{t-1})$. We iterate relation (6.4) with several layers, using layer normalization (Ba et al., 2016). We consider depth values from 1 to 6. We generate a dataset with $n = 2^{14}$ sequences with $T_{\max} = 50$ and $d = 5$ (therefore $e_t \in \mathbb{R}^{15}$) for training. We test using another dataset with 2^{10} sequences of the same shape. We train for 2000 epochs with Adam (Kingma and Ba, 2014) and a learning rate of 5×10^{-3} to minimize the mean squared error (MSE) $\min \ell(\Theta) \coloneqq \sum_{T=2}^{T_{\max}} \frac{1}{n} \sum_{i=1}^{n} \|\mathcal{T}_{\Theta}^{L}(e_{1:T}^{i}) - s_T^i\|^2$, where $\mathcal{T}_{\theta}^{L}$ correspond to L layers of (6.4) (we apply the forward rule L times, and then consider the section of first d coordinates). We compare the error with L steps of gradient descent on the inner loss (6.5), with a step size carefully chosen to obtain the fastest decrease. We find out that even though the first Transformer layers are competitive with gradient descent, the latter outperforms the Transformer by order of magnitudes when $L \geq 3$. Results are displayed in Figure 6.5. The fact that several steps of gradient descent outperform the same number of $\mathcal{T}_{\theta}$ layers is not surprising, as Proposition 6.5 does not generalize to more than one layer. In contrast, as shown in Appendix 6.B, a full Transformer with all the bells and whistles as described in Vaswani et al. (2017) (softmax and MLP applied component-wise to each Transformer layer) outperforms gradient descent and has a similar trend, as shown in Figure 6.10.

Non-Augmented setting. We now investigate whether the results of §6.5 still hold without assumptions 6.2 and 2. We consider the model $\mathcal{T}_{\theta}$ in (6.6). We parameterize the positional encoding in the linear Transformer equation (6.6) using the softmax of a positional attention-only similarity cost matrix with learnable parameters $W_{Q_{\text{pos}}}$ and $W_{K_{\text{pos}}}$: $P_{t,t'} = \text{softmax}(\langle W_{Q_{\text{pos}}} p_t | W_{K_{\text{pos}}} p_{t'} \rangle)$, as we found it to stabilize the training process. We use a similar dataset as in the previous section, i.e., a training set with 2^{14} sequences, each with 50 elements of dimension $d = 10$, and we test using another dataset with 2^{10} sequences of the same shape.

We train models $\mathcal{T}_{\theta}$ in (6.6) for 200 epochs with different numbers of heads. We use the Adam optimizer with a learning rate of 10^{-2}. Without further modification, we do not observe a significant gain as the number of heads increases. However, when duplicating the data along the dimension axis, that is $e_t \coloneqq (s_t, s_t)$, we observe a significant improvement, as illustrated in Figure 6.7. Understanding why duplicating the data leads to a significant improvement is left for future work.

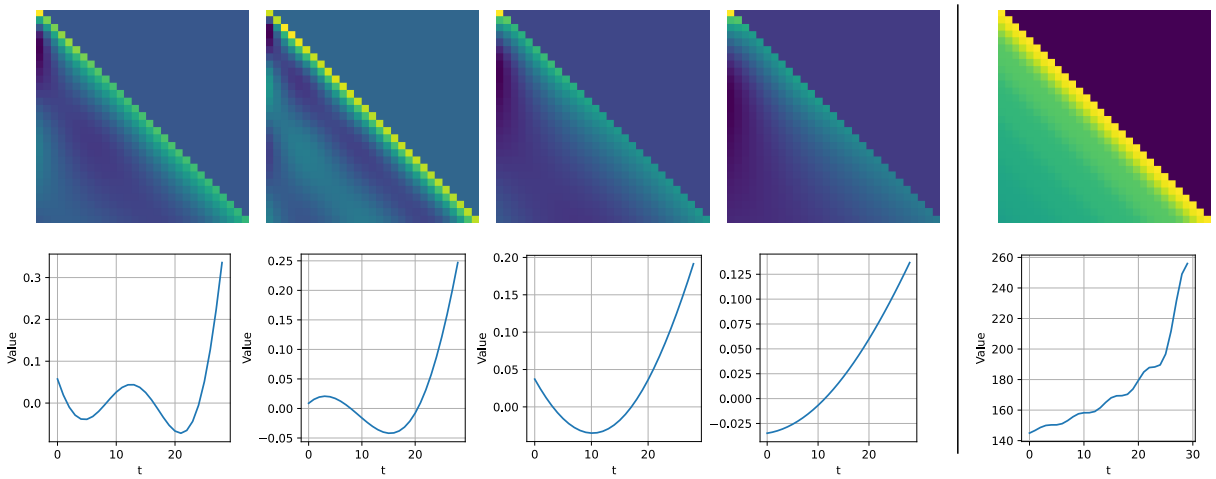

Figure 6.6: **Left:** Positional encodings after training for $\mu \in \{50, 100, 200, 300\}$. The first raw corresponds to the matrix P, and the second raw to a plot of its last raw. **Right:** Comparison with the cosine absolute positional encoding standardly used in machine translation Vaswani et al. (2017) (we display $pp^\top$). In both cases, we observe an invariance across diagonals. In addition, for high μ (i.e. small variations of the context), the most recent tokens are more informative, as imposed by the inductive prior of the cosine positional encoding.

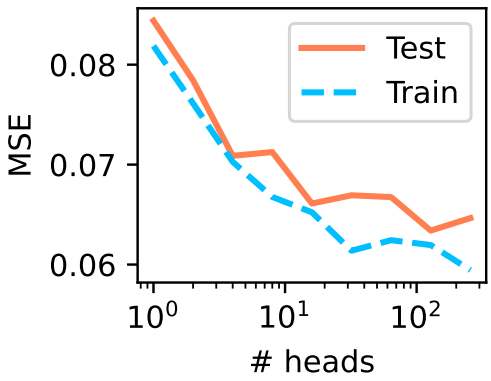

Figure 6.7: **Evolution** of the MSE with the number of heads. At initialization, the MSE is between 0.35 and 1.

To further relate our experimental findings to our theory, we also exhibit an orthogonality property between heads after training. For this, we take $d = 5$ to ease the visualization, and initialize each parameter equally across heads but add a small perturbation $.05 \times \mathcal{N}(0, 1)$ to ensure different gradient propagation during training. We then train the model and compare the quantity $(\sum_{h=1}^{H}(B_h^\top B_h))_{(i,j)}$ after training and at initialization. Note that it corresponds to a measure of orthogonality of heads. Results are displayed in Figure 6.8. We observe that after training, an orthogonality property appears. In addition, as we are duplicating the tokens across dimensions, we can see that heads become specialized in attending to some coordinates across tokens.

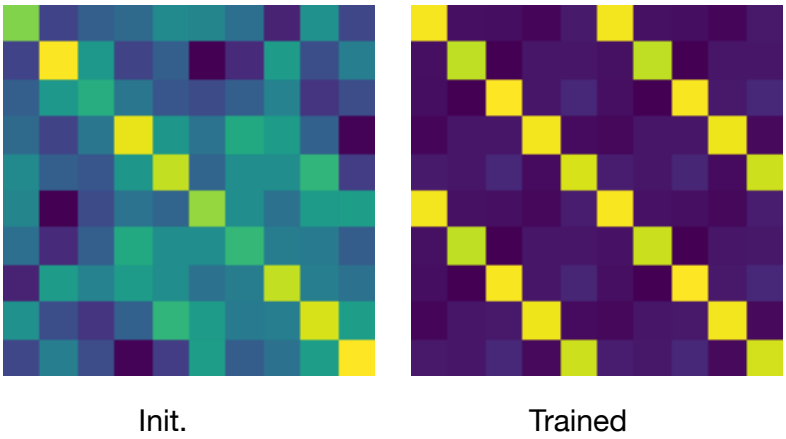

Figure 6.8: **Matrices** $(\sum_{h=1}^{H}(B_h^\top B_h))_{(i,j)} \in \mathbb{R}^{10\times 10}$ at initialization and after training. The trained parameters lead to an orthogonality between heads, as predicted by our theory.

Change in the context distribution. We consider the setting of §6.5.3, using the empirical loss counterpart of (6.7), averaged over $T \in \{2, \cdots, T_{\max}\}$. We generate a dataset with 10^4 examples and $T_{\max} = 30$. We train our positional encoding-only model with gradient descent and stop training (early stopping) when the loss is smaller than 10^{-3}. We initialize $P_{t,t'} = 0$. Results are in Figure 6.6, where we mask coefficients $P_{T-1,T}$ (which are close to 1 after training) in the display to investigate the behavior of the extra coefficients. We observe that the trained positional encoding exhibits an invariance across diagonals. Importantly, each row P_t has a smooth behavior with t', that we compare to absolute cosine positional encodings (Vaswani et al., 2017).

Conclusion

In this work, we study the in-context autoregressive learning abilities of linear Transformers to learn autoregressive processes of the form $s_{t+1} = Ws_t$. In-context autoregressive learning is decomposed into two steps: estimation of W with an in-context map Γ, followed by a prediction map ψ. Under commutativity and parameter structure assumptions, we first characterized Γ and ψ on augmented tokens, in which case Γ is a step of gradient descent on an inner objective function. We also considered non-augmented tokens and showed that Γ corresponds to a non-trivial geometric relation between tokens, enabled by an orthogonality between trained heads and learnable positional encoding. We also studied positional encoding-only attention and showed that approximate solutions of minimum ℓ_2 norm are favored by the optimization. Moving beyond commutativity assumptions, we extended our theoretical findings to the general case through experiments.

Future work. Investigating the case where $\tau = T$ in the non-augmented setting would lead to approximated in-context mappings, where achieving zero loss is no longer possible. This investigation would provide further insight into the role of positional encoding in estimating W in-context. Another investigation left for future work is to consider the case of non-commuting context matrices and to relate the computation of a Transformer to a proxy for a gradient flow for estimating W in-context, using the connection between Transformers and gradient flows Sander et al. (2022a).

6.A Proofs

In what follows, as we consider complex numbers, we use the hermitian product over $\mathbb{C}^d$, that is $\langle \alpha, \beta \rangle := \beta^\star \alpha = \sum_i \alpha_i \bar{\beta}_i$.

We denote $\mathcal{U}^d = \{\lambda \in \mathbb{C}^d | |\lambda_i| = 1 \forall i \in \{1, \cdots, d\}\}$.

6.A.1 Proof of Lemma 6.3.

Proof. Let, for $s \in \mathbb{R}^d$, $W_V^1 s := (0, s, 0) \in \mathbb{R}^{3d}$ and $W_V^2 s := (0, 0, s)$.

We now simply consider a positional attention-only model, that is, for $h \in \{1, 2\}$:

$$\mathcal{A}^h_{t,:} = \text{softmax}(P^h_{t,:}).$$

We can choose the positional encodings P^1 and P^2 such that $A^1_{t,t'} \simeq \delta_{t'=t}$ and $A^2_{t,t'} \simeq \delta_{t'=t-1}$.

Then

$$\sum_{h=1}^{2} \sum_{t'=1}^{t} \mathcal{A}^h_{t,t'} W_V^h s_{t'} \simeq (0, s_t, s_{t-1}) = e_t.$$

□

6.A.2 Proof of Proposition 6.4.

Proof. We briefly recall the reasoning as presented in Von Oswald et al. (2023b), and consider the case $W_0 = 0$ for simplicity. See Von Oswald et al. (2023b) for a full proof. Let

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & I_d & 0 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 0 & \eta I_d & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{6.8}$$

Then the section vector of the first d coordinates in (6.4) is $\eta \sum_{t=1}^{T} s_t s_{t-1}^\top s_T$.

The gradient of L at $W_0 = 0$ is:

$$\nabla_W L(0, e_{1:T}) = -\sum_{t=1}^{T-1} (s_{t+1}) s_t^\top.$$

Therefore, (6.4) corresponds to a single step of gradient descent starting from $W_0 = 0$. □

6.A.3 Proof of Proposition 6.5.

In what follows, the sums over t are from $t = 1$ to T and the sums over j are from $j = 1$ to d.

We first consider the following Lemma.

Lemma 6.13. *Under assumptions 6.2 and 1, the loss of the linear Transformer writes:*

$$\ell((a_i), (b_i)) = \mathbb{E}_\lambda \sum_{T=2}^{T_{\max}} \sum_{i=1}^{d} | \sum_{t,j,\alpha \in \mathcal{A}, \beta \in \mathcal{B}} c_{\alpha,\beta} \lambda_j^{T-t-\alpha} \lambda_i^{t-1+\beta} - \lambda_i^T |^2 \tag{6.9}$$

with $\mathcal{A} = \{-1, 0, 1\}$ and $\mathcal{B} = \{-1, 0\}$. We have $c_{\alpha,\beta} = u_\alpha v_\beta$ for $u_0 = a_1 + a_4$, $u_1 = a_2$, $u_{-1} = a_3$, $v_0 = b_1$ and $v_{-1} = b_2$.

Proof. One has $Ae_T = (0, A_1 s_T + A_2 s_{T-1}, A_3 s_T + A_4 s_{T-1})$. Therefore one has $\langle Ae_T, e_t \rangle = s_t^\star A_1 s_T + s_t^\star A_2 s_{T-1} + s_{t-1}^\star A_3 s_T + s_{t-1}^\star A_4 s_{T-1}$. Since $(Be_t)_{1:d} = B_1 s_t + B_2 s_{t-1}$, one obtains through (6.4):

$$\mathcal{T}_\theta(e_{1:T}) = \sum_t (a_1 s_t^\star s_T + a_2 s_t^\star s_{T-1} + a_3 s_{t-1}^\star s_T + a_4 s_{t-1}^\star s_{T-1})(b_1 s_t + b_2 s_{t-1}).$$

Developing, we obtain

$$\mathcal{T}_\theta(e_{1:T})_i = \sum_{t,j,\alpha \in \mathcal{A}} u_\alpha \lambda_j^{T-t-\alpha} \sum_{\beta \in \mathcal{B}} v_\beta \lambda_i^{t-1+\beta} = \sum_{t,j,\alpha \in \mathcal{A}, \beta \in \mathcal{B}} c_{\alpha,\beta} \lambda_j^{T-t-\alpha} \lambda_i^{t-1+\beta},$$

which implies the result. □

Using the notations of Lemma 6.13, Proposition 6.5 now writes as follows.

Proposition 6.14 (In-context autoregressive learning with gradient-descent.)**.** *Suppose $\mathcal{C} = \mathcal{C}_U$, assumptions 6.2 and 1. Then loss (6.9) is minimal for $c_{\alpha,\beta} = 0$ if $(\alpha, \beta) \neq (-1, 0)$ and $c_{-1,0} = \frac{\sum_{T=2}^{T_{\max}} T}{\sum_{T=2}^{T_{\max}} (T^2 + (d-1)T)}$. Therefore, the optimal in-context map Γ_{θ^*} is one step of gradient descent starting from the initialization $\lambda = 0$, with a step size asymptotically equivalent to $\frac{3}{2T_{\max}}$ with respect to $T_{\max}$.*

Proof. We develop the term in the sum in (6.9):

$$\begin{aligned}
&| \sum_{t,j,\alpha \in \mathcal{A}, \beta \in \mathcal{B}} c_{\alpha,\beta} \lambda_j^{T-t-\alpha} \lambda_i^{t+\beta-1} - \lambda_i^T |^2 \\
&= \sum_{t_1,j_1,\alpha_1,\beta_1,t_2,j_2,\alpha_2,\beta_2} c_{\alpha_1,\beta_1} c_{\alpha_2,\beta_2} \lambda_{j_1}^{T-t_1-\alpha_1} \lambda_{j_2}^{-T+t_2+\alpha_2} \lambda_i^{t_1-t_2+\beta_1-\beta_2} \\
&- 2\mathrm{Real}(\sum_{t,j,\alpha \in \mathcal{A}, \beta \in \mathcal{B}} c_{\alpha,\beta} \lambda_j^{T-t-\alpha} \lambda_i^{t+\beta-1} \lambda_i^{-T}) + 1.
\end{aligned}$$

We now need to compute the expectations of the first two terms in the sum. Because $\mathbb{E}_\lambda(\lambda_i^k) = \delta_{k=0}$ and the λ_i are i.i.d., most terms will be zeros.

- For the first term, one needs to look at the different possible values for (j_1, j_2, i) to calculate $\mathbb{E}_\lambda(\lambda_{j_1}^{T-t_1-\alpha_1} \lambda_{j_2}^{-T+t_2+\alpha_2} \lambda_i^{t_1-t_2+\beta_1-\beta_2})$.

  - 1) If $j_1 = j_2 = i$, it is $\mathbb{E}_\lambda(\lambda_i^{\alpha_2-\alpha_1+\beta_1-\beta_2}) = \delta_{\alpha_1-\alpha_2=\beta_1-\beta_2}$.

  - 2) If $j_1 = j_2 \coloneqq j \neq i$, it is $\mathbb{E}_\lambda(\lambda_j^{t_2-t_1+\alpha_2-\alpha_1}) \mathbb{E}_\lambda(\lambda_i^{t_1-t_2+\beta_1-\beta_2}) = \delta_{t_2-t_1=\alpha_1-\alpha_2=\beta_1-\beta_2}$.

  - 3) If $j_1 \neq j_2$, $i \neq j_1$, $i \neq j_2$, then $\mathbb{E}_\lambda(\lambda_{j_1}^{T-t_1-\alpha_1} \lambda_{j_2}^{-T+t_2+\alpha_2} \lambda_i^{t_1-t_2+\beta_1-\beta_2}) = \delta_{t_1-t_2=\beta_2-\beta_1, T-t_1-\alpha_1=0, T-t_2-\alpha_2=0}$

  - 4) If $j_1 \neq j_2$ and $i = j_1$, the expectation is $\delta_{T-\alpha_1-t_2+\beta_1-\beta_2=0, T=t_2+\alpha_2}$, and similarly when $j_1 \neq j_2$ and $i = j_2$.

  As a consequence, we see that all the terms that do not satisfy $\alpha_1 - \alpha_2 = \beta_1 - \beta_2$ will lead to 0 expectation, which therefore implies that the first term writes:

$$\mathbb{E}_\lambda(\sum_{t_1,j_1,\alpha_1,\beta_1,t_2,j_2,\alpha_2,\beta_2,\alpha_1-\alpha_2=\beta_1-\beta_2} c_{\alpha_1,\beta_1} c_{\alpha_2,\beta_2} \lambda_{j_1}^{T-t_1-\alpha_1} \lambda_{j_2}^{-T+t_2+\alpha_2} \lambda_i^{t_1-t_2+\beta_1-\beta_2}).$$

- For the second term, we have

$$\sum_{t,j,\alpha\in\mathcal{A},\beta\in\mathcal{B}} c_{\alpha,\beta}\lambda_j^{T-t-\alpha}\lambda_i^{t+\beta-1}\lambda_i^{-T} = \sum_{t,\alpha,\beta} c_{\alpha,\beta}\sum_j(\lambda_j^{T-t-\alpha}\lambda_i^{t+\beta-1-T}).$$

  - When $i \neq j$, the expectation of $\lambda_j^{T-t-\alpha}\lambda_i^{t+\beta-1-T}$ is not 0 if and only if $t+\alpha = T$ and $T = t+\beta-1$. Given that $\mathcal{A} = \{-1,0,1\}$ and $\mathcal{B} = \{-1,0\}$, this implies $\alpha = \beta - 1$ and therefore $\alpha = -1$ and $\beta = 0$. But then we have $t = T+1$ which is not possible.

  - When $i = j$, the expectation of $\lambda_j^{T-t-\alpha}\lambda_i^{t+\beta-1-T} = \lambda_i^{\beta-\alpha-1}$ is not 0 if and only if $\beta = \alpha+1$, that is $\beta = 0$ and $\alpha = -1$.

  Therefore, one has

$$\mathbb{E}_\lambda(\sum_{t,\alpha,\beta} c_{\alpha,\beta}\sum_j(\lambda_j^{T-t-\alpha}\lambda_i^{t+\beta-1-T})) = \sum_t c_{-1,0} = Tc_{-1,0}$$

  and the second term is $-2Tc_{-1,0}$.

Back to the full expectation, isolating the term in $c_{-1,0}^2$ from the first term, we get

$$\mathbb{E}_\lambda|\sum_{t,j,\alpha\in\mathcal{A},\beta\in\mathcal{B}} c_{\alpha,\beta}\lambda_j^{T-1-t-\alpha}\lambda_i^{t+\beta-1} - \lambda_i^T|^2 = \tag{6.10}$$

$$\mathbb{E}_\lambda(\sum_{t_1,j_1,\alpha_1,\beta_1,t_2,j_2,\alpha_2,\beta_2,\alpha_1-\alpha_2=\beta_1-\beta_2,(\alpha_1,\alpha_2,\beta_1,\beta_2)\neq(-1,-1,0,0)} c_{\alpha_1,\beta_1}c_{\alpha_2,\beta_2}\lambda_{j_1}^{T-t_1-\alpha_1}\lambda_{j_2}^{-T+t_2+\alpha_2}\lambda_i^{t_1-t_2+\beta_1-\beta_2})$$

$$+K_Tc_{-1,0}^2 - 2Tc_{-1,0} + 1,$$

for some constant K_T.

We now examine the terms in the sum within the expectation. We want to show that $E_1 = E_2$ where

$$E_1 \coloneqq \{(\alpha_1,\alpha_2,\beta_1,\beta_2)|\alpha_1-\alpha_2 = \beta_1-\beta_2, (\alpha_1,\alpha_2,\beta_1,\beta_2) \neq (-1,-1,0,0)\}$$

and

$$E_2 \coloneqq \{(\alpha_1,\alpha_2,\beta_1,\beta_2)|\alpha_1-\alpha_2 = \beta_1-\beta_2, (\alpha_1,\beta_1) \neq (-1,0), (\alpha_2,\beta_2) \neq (-1,0)\}.$$

We already have $E_2 \subset E_1$. If $(\alpha_1,\alpha_2,\beta_1,\beta_2) \in E_1\backslash E_2$, then either $(\alpha_1,\beta_1) = (-1,0)$ or $(\alpha_2,\beta_2) = (-1,0)$. If $(\alpha_1,\beta_1) = (-1,0)$, since $\alpha_1-\alpha_2 = \beta_1-\beta_2$, then $\alpha_2 = \beta_2 - 1$, which necessarily implies $\beta_2 = 0$ and $\alpha_2 = -1$, which contradicts the fact that $(\alpha_1,\alpha_2,\beta_1,\beta_2) \in E_1$. Similarly, if $(\alpha_2,\beta_2) = (-1,0)$ and $\alpha_1-\alpha_2 = \beta_1-\beta_2$, then $\beta_1 = 0$ and $\alpha_1 = -1$. Therefore, $E_1 = E_2$, and

$$\mathbb{E}_\lambda(\sum_{t_1,j_1,t_2,j_2,(\alpha_1,\alpha_2,\beta_1,\beta_2)\in E_1} c_{\alpha_1,\beta_1}c_{\alpha_2,\beta_2}\lambda_{j_1}^{T-t_1-\alpha_1}\lambda_{j_2}^{-T+t_2+\alpha_2}\lambda_i^{t_1-t_2+\beta_1-\beta_2}) =$$

$$\mathbb{E}_\lambda(\sum_{t_1,j_1,t_2,j_2,(\alpha_1,\alpha_2,\beta_1,\beta_2)\in E_2} c_{\alpha_1,\beta_1}c_{\alpha_2,\beta_2}\lambda_{j_1}^{T-t_1-\alpha_1}\lambda_{j_2}^{-T+t_2+\alpha_2}\lambda_i^{t_1-t_2+\beta_1-\beta_2}) =$$

$$\mathbb{E}_\lambda(|\sum_{t,j,\alpha\in\mathcal{A},\beta\in\mathcal{B},(\alpha,\beta)\neq(-1,0)} c_{\alpha,\beta}\lambda_j^{T-t-\alpha}\lambda_i^{t+\beta-1}|^2) \geq 0.$$

We are interested in the minimum of (6.10). The minimum of $K_T c_{-1,0}^2 - 2Tc_{-1,0} + 1$ is reached for $c_{-1,0} \neq 0$. We also just showed that the first term is non-negative, so the minimum will be reached when (almost surely in $\lambda \sim \mathcal{U}(\mathcal{C}_U)$)

$$\sum_{t,j,\alpha\in\mathcal{A},\beta\in\mathcal{B},(\alpha,\beta)\neq(-1,0)} c_{\alpha,\beta}\lambda_j^{T-t-\alpha}\lambda_i^{t+\beta-1} = 0.$$

In particular, the terms corresponding to monomials in λ_i are

$$\sum_{t,\alpha\in\mathcal{A},\beta\in\mathcal{B},(\alpha,\beta)\neq(-1,0)} c_{\alpha,\beta}\lambda_i^{T-\alpha+\beta-1} = (T \sum_{\alpha\in\mathcal{A},\beta\in\mathcal{B},(\alpha,\beta)\neq(-1,0)} c_{\alpha,\beta})\lambda_i^{T-\alpha+\beta-1} = 0.$$

Therefore, identifying the coefficients of this polynomial gives $c_{1,-1} = c_{-1,-1} + c_{0,0} = c_{1,0} + c_{0,-1} = 0$. We want to show that this implies $v_{-1} = u_1 = u_0 = 0$.

If $v_{-1} \neq 0$, then because $c_{1,-1} = 0$, we have $u_1 = 0$. From $c_{1,0} + c_{0,-1} = 0$, it follows $c_{0,-1} = 0$ and therefore $u_0 = 0$. From $c_{-1,-1} + c_{0,0} = 0$, this implies $c_{-1,-1} = 0$ and thus $u_{-1} = 0$, which contradicts $c_{-1,0} \neq 0$. Therefore, $v_{-1} = 0$.

Now, if $u_1 \neq 0$, then $v_{-1} = 0$, and from $c_{-1,-1} + c_{0,0} = 0$, we have $c_{0,0} = 0$. But because $c_{-1,0} \neq 0$, we have $u_0 = 0$, which combined with $c_{1,0} + c_{0,-1} = 0$ implies $c_{1,0} = 0$, which is impossible because $v_0 \neq 0$. Therefore, $u_1 = 0$.

So $u_1 = v_{-1} = 0$, and $c_{0,0} = 0$, so that $u_0 = 0$. This shows that loss (6.9) is minimal for $c_{\alpha,\beta} = 0$ if $(\alpha, \beta) \neq (-1, 0)$.

Last, we need to calculate the constant

$$K_T \coloneqq \mathbb{E}_\lambda(\sum_{t_1,j_1,t_2,j_2} \lambda_{j_1}^{T-t_1+1}\lambda_{j_2}^{-T+t_2-1}\lambda_i^{t_1-t_2}).$$

Back to the different possible values for (j_1, j_2, i) analyzed above, we get non-zeros when $j_1 = j_2 = i$ and when $j_1 = j_2 \neq i$. In cases 3) and 4) we obtain as a necessary condition for non-zero expectation: $t_2 = T + 1$ or $t_1 = T + 1$; which is not possible. Therefore,

$$K_T = (\sum_{j_1=j_2=i} \sum_{t_1,t_2} 1) + (\sum_{j_1=j_2\neq i} \sum_{t_1=t_2} 1) = T^2 + (d-1)T.$$

We now denote $\eta = a_3 b_1 = c_{-1,0}$. Replacing the zero terms in loss (6.9), we obtain

$$d \sum_{T=2}^{T_{\max}} \left(\eta^2(T^2 + (d-1)T) - 2\eta T + 1\right),$$

for which the argmin is given by

$$\eta^* = \frac{\sum_{T=2}^{T_{\max}} T}{\sum_{T=2}^{T_{\max}} (T^2 + (d-1)T)} \sim \frac{3}{2T_{\max}} \quad \text{as} \quad T_{\max} \to +\infty.$$

Finally at optimality,

$$\mathcal{T}_{\theta^*}(e_{1:T}) = \sum_t \eta^* s_{t-1}^\star s_T s_t = \eta^*(\sum_t s_t s_{t-1}^\star) s_T = -\eta^* \nabla_W L(0, e_{1:T}) s_T = \Gamma_{\theta^*}(e_{1:T}) s_T$$

with $\Gamma_{\theta^*}(e_{1:T}) \coloneqq -\eta^* \nabla_W L(0, e_{1:T})$. □

6.A.4 Proof of Lemma 6.6.

Proof. For a given input sequence $e_{1:T} = s_{1:T} = (s, \lambda \odot s, \cdots, \lambda^{T-1} \odot s)$, with context $\lambda \in \mathcal{U}^d$, one has

$$\mathcal{T}_\theta(e_{1:T}) = \sum_{t=1}^{T} P_{T-1,t} \sum_{h=1}^{H} \langle \lambda^{t-1} \odot s, a_h \odot \lambda^{T-2} \odot s \rangle b_h \odot \lambda^{t-1} \odot s.$$

We have

$$(\langle \lambda^{t-1} \odot s, a_h \odot \lambda^{T-2} \odot s \rangle b_h \odot \lambda^{t-1} \odot s)_i = \sum_{j=1}^{d} a_h^j \lambda_j^{2-T} \lambda_j^{t-1} b_h^i \lambda_i^{t-1}.$$

Since we precisely have

$$([\mathtt{B}^\top \mathtt{A}] \lambda^{t-T+1} \odot \lambda^{t-1})_i = \sum_{j=1}^{d} \sum_{h=1}^{H} b_h^i a_h^j \lambda_j^{t-T+1} \ \lambda_i^{t-1},$$

this gives us the desired result.

□

Remark. In fact, looking at the above proof, considering arbitrary real values for the vector $s_0 = s$, one can absorb the terms in s_j^2 in each a_h^j and the terms in s_i in each b_h^i, since these are learnable parameters. Therefore, our results can be adapted to any arbitrary initial value s_0.

6.A.5 Proof of Proposition 6.7.

Proof. Denote $\mathtt{C} = \mathtt{B}^\top \mathtt{A}$. Let us suppose that we have an optimal solution θ such that $l(\theta) = 0$. Therefore, for almost all $\lambda \in \mathcal{U}^d$ and $s_t = \lambda^{t-1}$, one has $\mathcal{T}_\theta(s_{1:T}) = \lambda^T$. Then, $\forall i \in \{1, \cdots, d\}$:

$$\sum_{t=1}^{T} P_{T-1,t} \sum_{j=1}^{d} \mathtt{C}_{ij} \lambda_j^{t-T+1} \lambda_i^{t-1} = \lambda_i^T.$$

By identifying the coefficients of the polynomial in the λ_i's, we see that one must have for all $T \geq 2$ that $P_{T-1,t} = 0$ if $t \neq T$, and for all $1 \leq i \leq d$, $p_{T-1,T} \mathtt{C}_{ii} = 1$, and $\mathtt{C}_{ij} = 0$ for $i \neq j$.

An optimal in-context mapping is then obtained by considering the forward rule of $\mathcal{T}_{\theta^*}$ for optimal parameters. One gets:

$$\mathcal{T}_{\theta^*}(e_{1:T}) = (\bar{e}_{T-1} \odot e_T) \odot (e_T).$$

□

6.A.6 Proof of Proposition 6.8.

Proof. The loss writes

$$\ell(\theta) = \sum_{T=2}^{T_{\max}} \mathbb{E}_{\operatorname{diag} \lambda \sim \mathcal{W}} \| \sum_{t=1}^{T} P_{T-1,t} \mathtt{C} \lambda^{t-T+1} \odot \lambda^{t-1} - \lambda^T \|^2.$$

We compute the expectation of each term in the sum by first developing it.

One has

$$\|\sum_{t=1}^{T} P_{T-1,t}\mathtt{C}\lambda^{t-T+1}\odot\lambda^{t-1}-\lambda^{T}\|^2=\sum_{t,t'}P_{T-1,t}P_{T-1,t'}\sum_{i,j,k}\mathtt{C}_{ij}\mathtt{C}_{ik}\lambda_j^{T-t-1}\lambda_k^{t'-T+1}\lambda_i^{t'-t}$$

$$-2\mathrm{Real}(\sum_{t}P_{T-1,t}\sum_{i,j}\mathtt{C}_{ij}\lambda_j^{T-t-1}\lambda_i^{T-t+1})+d.$$

Looking at the expectation of the first term

$$\sum_{t,t'}P_{T-1,t}P_{T-1,t'}\sum_{i,j,k}\mathtt{C}_{ij}\mathtt{C}_{ik}\mathbb{E}(\lambda_j^{T-t-1}\lambda_k^{t'-T+1}\lambda_i^{t'-t}),$$

we see that one has to calculate for t,t',i,j,k

$$\mathbb{E}(\lambda_j^{T-t-1}\lambda_k^{t'-T+1}\lambda_i^{t'-t}).$$

When $j\neq k$, it is $\delta_{t'=t=T-1}$. When $j=k$ it is $\delta_{t'=t}$.

Therefore

$$\sum_{t,t'}P_{T-1,t}P_{T-1,t'}\sum_{i,j,k}\mathtt{C}_{ij}\mathtt{C}_{ik}\mathbb{E}(\lambda_j^{T-t-1}\lambda_k^{t'-T+1}\lambda_i^{t'-t})=\sum_{t=t'}P_{T-1,t}^2\sum_{i,j}\mathtt{C}_{ij}^2+P_{T-1,T-1}^2\sum_{i,j,k}\mathtt{C}_{i,j}\mathtt{C}_{i,k}=$$

$$\|P_{T-1}\|^2\|\mathtt{C}\|_F^2+P_{T-1,T-1}^2S(\mathtt{C}^\top\mathtt{C}).$$

Similarly, because $\mathbb{E}(\lambda_j^{T-t-1}\lambda_i^{T-t+1})=\delta_{t=T,i=j}$, the second term is

$$-2\mathbb{E}(\mathrm{Real}(\sum_{t}P_{T-1,t}\sum_{i,j}\mathtt{C}_{ij}\lambda_j^{T-t-1}\lambda_i^{T-t+1}))=-2P_{T-1,T}\mathrm{Tr}(\mathtt{C}).$$

This concludes the proof. □

6.A.7 Proof of Proposition 6.9.

Proof. Let us denote $a(t)$, $b(t)$ and $p(t)$ the functions defined by the gradient flow on loss ℓ, that is $\dot{a}=-\nabla_a\ell(a,b,p)$ and similarly for b and p. We start with the result from Nguegnang et al. (2021); Achour et al. (2021), which states that $a(t)$, $b(t)$, and $p(t)$ are bounded and converge to a stationary point (a^*,b^*,p^*) as $t\to\infty$:

$$\lim_{t\to\infty}(a(t),b(t),p(t))=(a^*,b^*,p^*).$$

The possible stationary points are either global or non-global minima. The conditions for these are:

- Global minimum if $p^*a^*b^*=1$
- Non-global minimum if $a^*b^*=0$, $a^*p^*=0$ and $b^*p^*=0$ and therefore

$$p^*a^*b^*=0.$$

If the system were to converge to a non-global minimum, we would therefore have

$$\ell(+\infty)=|p^*a^*b^*-1|=|0-1|=1>\ell(0)$$

This is a contradiction because the energy should have decreased over time.

As a result, the only remaining possibility for the stationary points is that they must satisfy $p^*a^*b^*=1$. Therefore, the functions $a(t)$, $b(t)$, and $p(t)$ must converge to a global minimum. □

6.A.8 Proof of Lemma 6.10.

Proof. For the announced parameters, the problem simply decomposes into sub-problems in dimension 2. Indeed, we have for all $i \in \{1, \dots, \delta\}$:

$$(\mathcal{T}_{\theta^*}(e_{1:T}))_{2i-1} = -\frac{1}{2}(\lambda_{2i-1}^0 + \bar{\lambda}_{2i-1}^0)\lambda_{2i-1}^{T-2} + (\lambda_{2i-1}^1 + \bar{\lambda}_{2i-1}^1)\lambda_{2i-1}^{T-1} = \lambda_{2i-1}^{T-2}(-1 + \lambda_{2i-1}^2 + 1) = \lambda_{2i-1}^T.$$

Similarly,

$$(\mathcal{T}_{\theta^*}(e_{1:T}))_{2i} = \lambda_{2i}^T.$$

□

6.A.9 Proof of Proposition 6.11.

Proof. The proof is similar to Proposition 6.A.5, by regrouping each λ_i with $\bar{\lambda}_i$ and identifying coefficients in two polynomials. More precisely, one must have

$$\sum_{t=1}^{T} P_{T-1,t} \sum_{j=1}^{d} \mathtt{C}_{2i-1,j} \lambda_j^{t-T+1} \lambda_{2i-1}^{t-1} = \lambda_{2i-1}^T.$$

Therefore, isolating terms in λ_{2i-1} (recall that $\lambda_{2i} = 1/\lambda_{2i-1}$), and developing, we get, noting $p \coloneqq P_{T-1}$:

$$\mathtt{C}_{2i-1,2i-1}\Big(\sum_{t<T-1} p_t \lambda_{2i-1}^{2t-T} + p_{T-1}\lambda_{2i-1}^{T-2} + p_T \lambda_{2i-1}^T\Big) + \sum_t p_t \mathtt{C}_{2i-1,2i} \lambda_{2i-1}^{T-2} = \lambda_{2i-1}^T.$$

Identifying gives $(\mathtt{C}_{2i-1,2i-1} + \mathtt{C}_{2i-1,2i})p_{T-1} + \mathtt{C}_{2i-1,2i}p_T = 0$, $\mathtt{C}_{2i-1,2i-1}p_T = 1$ and $p_{t<T-1} = 0$.

Similarly, on the conjugates, $(\mathtt{C}_{2i,2i} + \mathtt{C}_{2i,2i-1})p_{T-1} + \mathtt{C}_{2i,2i-1}p_T = 0$, $\mathtt{C}_{2i,2i}p_T = 1$.

Identifying the other terms gives $C_{2i,j} = C_{2i-1,j} = 0$ for $j \neq 2i$ and $j \neq 2i-1$. □

Interpretation. Up to rescaling, the relation $p_T C_{i,i} = 1$ gives $p_T = C_{i,i} = 1$, and therefore

$$C_{2i-1,2i} + (1 + C_{2i-1,2i})p_{T-1} = C_{2i,2i-1} + (1 + C_{2i,2i-1})p_{T-1} = 0, \tag{6.11}$$

which gives $C_{2i-1,2i} = C_{2i,2i-1}$. Therefore, C is symmetric and has the form $\mathrm{diag}(J_b, \dots, J_b)$ with $J_b = ((1, b), (b, 1))$ and $b = -p_{T-1}/(1 + p_{T-1})$. If $H < d$, C cannot be full rank, and therefore necessarily $C_{2i,2i-1} = 1$ or -1. But $C_{2i,2i-1} = -1$ is impossible given (6.11). We then recover Lemma 6.10.

6.A.10 Proof of Proposition 6.12

Proof. One has that the second order term in the quadratic form (6.7) writes

$$\langle p | Hp \rangle = \sum_{t,t'} p_t p_{t'} \mathbb{E}_{\lambda \sim \mathcal{W}(\mu)} \lambda^{2(t'-t)}.$$

We can calculate this expectation in closed form. Writing $\alpha = \frac{2\pi}{\mu}$, it gives

$$\mathbb{E}_{\lambda \sim \mathcal{W}(\mu)} \lambda^{2(t'-t)} = \frac{1}{\alpha} \int_0^{\alpha} e^{2i\theta(t'-t)} d\theta,$$

which gives $\frac{i}{2\alpha(t'-t)}[1 - e^{2i(t'-t)\alpha}]$ if $t \neq t'$ and 1 otherwise. Since $\langle p|Hp\rangle$ is real, we can identify the real parts so that

$$H_{t,t'} = (\mu/(4\pi(t'-t))) \sin(4(t'-t)\frac{\pi}{\mu}).$$

We now turn to the eigenvalues of H. H is a smooth function of α:

$$H : [0, 2\pi] \to S_T,$$

where S_T are the symmetric matrices of $\mathbb{R}^{T \times T}$. Using Th. 5.2 from Kato (2013), we know that the eigenvalues of H can be parametrized as continuous functions $\nu_1(\alpha) \geq \nu_2(\alpha) \geq \cdots \nu_T(\alpha)$. Since for $\alpha = 0$, the eigenvalues of H are $T, 0, \cdots 0$, and recalling that $\mu = \frac{2\pi}{\alpha}$, we obtain the result.

□

6.B Additional Experiments

Effect of the addition of softmax **layers and MLP layers on the trained solutions.** We consider the unitary context matrix setup of section 6.5. As mentionned, we could not find a natural way to express the global minimum of the training loss when a softmax layer was involved, even in dimension 1. To get more insight, we conducted an additional experiment where we trained different models with and without softmax and MLP layers. We used $d = 1$, and $T = 10$, and a hidden dimension of 32 in the MLP. Results are shown in Figure 6.9, where it is clear that in the case of commuting context matrices, using a softmax is incompatible with learning the underlying in-context mapping.

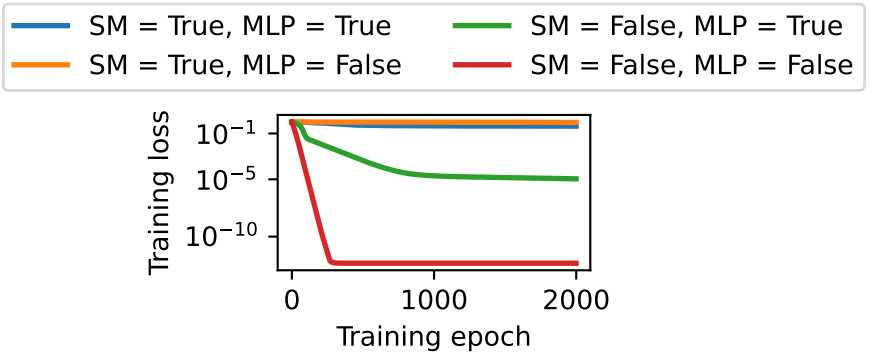

Figure 6.9: Training loss with training epoch for different configurations of the Transformer's architecture, when using a softmax (SM) and an MLP or not.

Transformer with all the bells and whistles and gradient descent. The experiment displayed in Figure 6.5 shows that linear attention fails to compete with gradient descent when there are more than 2 layers. In contrast, a full Transformer with all the bells and whistles as described in Vaswani et al. (2017) (softmax and MLP applied component-wise to each transformer layer) outperforms gradient descent and has a similar trend, as shown in Figure 6.10. The training procedure and the dataset are identical to those described in the *Augmented setting* paragraph of Section 6.6.

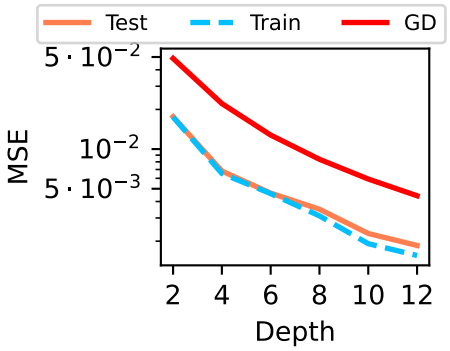

Figure 6.10: **Evolution** of the mean squared errors (MSE) with depth L for a vanilla Transformer Vaswani et al. (2017). We compare with L steps of gradient descent (GD) on the inner loss (6.5). At initialization, the MSE is between 1 and 2. .

7

Fast, Differentiable and Sparse Top-k: a Convex Analysis Perspective

The top-k operator returns a sparse vector, where the non-zero values correspond to the k largest values of the input. Unfortunately, because it is a discontinuous function, it is difficult to incorporate in neural networks trained end-to-end with backpropagation. Recent works have considered differentiable relaxations, based either on regularization or perturbation techniques. However, to date, no approach is fully differentiable *and* sparse. In this chapter, we propose new differentiable *and* sparse top-k operators. We view the top-k operator as a linear program over the permutahedron, the convex hull of permutations. We then introduce a p-norm regularization term to smooth out the operator, and show that its computation can be reduced to isotonic optimization. Our framework is significantly more general than the existing one and allows for example to express top-k operators that select values *in magnitude*. On the algorithmic side, in addition to pool adjacent violator (PAV) algorithms, we propose a new GPU/TPU-friendly Dykstra algorithm to solve isotonic optimization problems. We successfully use our operators to prune weights in neural networks, to fine-tune vision transformers, and as a router in sparse mixture of experts.

Contents

7.1 Introduction

Finding the top-k values and their corresponding indices in a vector is a widely used building block in modern neural networks. For instance, in sparse mixture of experts (MoEs) (Shazeer et al., 2017; Fedus et al., 2022), a top-k router maps each token to a selection of k experts (or each expert to a selection of k tokens). In beam search for sequence decoding (Wiseman and Rush, 2016), a beam of k possible output sequences is maintained and updated at each decoding step. For pruning neural networks, the top-k operator can be used to sparsify a neural network, by removing weights with the smallest magnitude (Han et al., 2015; Frankle and Carbin, 2018). Finally, top-k accuracy (e.g., top-3 or top-5) is frequently used to evaluate the performance of neural networks at inference time.

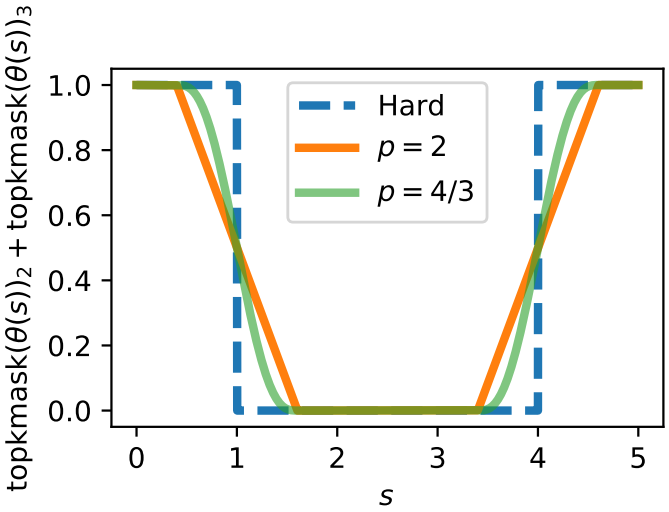

Figure 7.1: **Illustration of our differentiable and sparse top-k mask.** For $k = 2$, we consider $\boldsymbol{\theta}(s) = (3, 1, -1 + s, s) \in \mathbb{R}^4$ and plot $\text{topkmask}(\boldsymbol{\theta}(s))_2 + \text{topkmask}(\boldsymbol{\theta}(s))_3$ as a function of s. We compare the hard version (no regularization) with our proposed operator using p-norm regularization: $p = 2$ leads to differentiable a.e. operator; $p = 4/3$, leads to a differentiable operator. Both operators are **sparse**: they are exactly 0 for some values of s.

However, the top-k operator is a discontinuous piecewise affine function with derivatives either undefined or constant (the related top-k *mask* operator, which returns a binary encoding of the indices corresponding to the top-k values, has null derivatives). This makes it hard to use in a neural network trained with gradient backpropagation. Recent works have considered

differentiable relaxations, based either on regularization or perturbation techniques (see §7.2 for a review). However, to date, no approach is differentiable everywhere *and* sparse. Sparsity is crucial in neural networks that require *conditional* computation. This is for instance the case in sparse mixture of experts, where the top-k operator is used to "route" tokens to selected experts. Without sparsity, all experts would need to process all tokens, leading to high computational cost. This is also the case when selecting weights with highest magnitude in neural networks: the non-selected weights should be exactly 0 in order to optimize the computational and memory costs.

In this work, we propose novel differentiable everywhere *and* sparse top-k operators (see Figure 7.1). We build upon the framework of Blondel et al. (2020b), which casts sorting and ranking as linear programs over the permutahedron, and uses a reduction to isotonic optimization. We significantly generalize that framework in several ways. Specifically, we make the following contributions:

- After reviewing related work in §7.2 and background in §7.3, we introduce our generalized framework in §7.4. We introduce a new nonlinearity φ, allowing us to express new operators (such as the top-k in magnitude). In doing so, we also establish new connections between so-called k-support norms and the permutahedron.
- We introduce a regularization term to obtain a relaxed top-k operator. In particular, using p-norm regularization, we obtain the first differentiable everywhere *and* sparse top-k operator (Figure 7.3).
- In §7.5, we derive pool adjacent violator (PAV) algorithms for solving isotonic optimization when using φ and/or when using p-norm regularization. We show that the Jacobian of our operator can be computed in closed form.
- As a GPU/TPU friendly alternative to PAV, we propose a Dykstra algorithm to solve isotonic optimization, which is easy to vectorize in the case $p = 2$.
- In §7.6, we chose to focus on three applications of our operators. First, we use them to prune weights in a multilayer perceptron during training and show that they lead to better accuracy than with a hard top-k. Second, we define top-k losses to fine-tune vision transformers (ViTs) and obtain better top-k accuracy than with the cross-entropy loss. Finally, we use our operators as a router in vision mixture of experts, and show that they outperform the hard top-k router.

7.2 Related work

Differentiable loss functions. Several works have proposed a differentiable loss function as a surrogate for a discrete, discontinuous metric. For example, loss functions have been proposed for top-k accuracy (Lapin et al., 2015, 2016; Berrada et al., 2018; Petersen et al., 2022) and various ranking metrics (Chapelle and Wu, 2010; Adams and Zemel, 2011; Rolínek et al., 2020).

Differentiable operators. In a different line of research, which is the main focus of this work, a differentiable operator is proposed, which can be used either as an intermediate layer in a neural network or as final output, fed into an arbitrary loss function. For example, Niepert et al. (2021) proposed a framework for computing gradients of discrete probability distributions or optimization problems. Amos et al. (2019) and Qian et al. (2022) proposed a smooth (but not sparse) top-k operator based on binary entropy regularization. Related projections on the capped simplex were proposed (Martins and Kreutzer, 2017; Malaviya et al., 2018; Blondel, 2019) in different contexts. These works use ad-hoc algorithms, while we use a reduction to isotonic optimization. Cuturi et al. (2019) proposed a relaxation of the sorting and ranking operators

based on entropy-regularized optimal transport and used it to obtain a differentiable (but again not sparse) top-k operator. Its computation relies on Sinkhorn's algorithm (Sinkhorn, 1967; Cuturi, 2013), which in addition makes it potentially slow to compute and differentiate. A similar approach was proposed by Xie et al. (2020). Petersen et al. (2021) propose smooth differentiable sorting networks by combining differentiable sorting functions with sorting networks. Other relaxation of the sort and operators have been proposed by Grover et al. (2019) and Prillo and Eisenschlos (2020).

The closest work to ours is that of Blondel et al. (2020b), in which sorting and ranking are cast as linear programs over the permutahedron. To make these operators differentiable, regularization is introduced in the formulation and it is shown that the resulting operators can be computed via isotonic optimization in $O(n \log n)$ time. Unfortunately, the proposed operators still include kinks: they are not differentiable everywhere. The question of how to construct a differentiable everywhere *and* sparse relaxation with $O(n \log n)$ time complexity is therefore still open. In this work, we manage to do so by using p-norm regularization. Furthermore, by introducing a new nonlinearity φ, we significantly generalize the framework of Blondel et al. (2020b), allowing us for instance to express a new top-k operator *in magnitude*. We introduce a new GPU/TPU friendly Dykstra algorithm as an alternative to PAV.

Instead of introducing regularization, another technique relies on perturbation (Berthet et al., 2020). This technique has been used to obtain a differentiable (but still *not sparse*) top-k for image patch selection (Cordonnier et al., 2021).

Pruning weights with small magnitude. Many recent works focus on neural network pruning, where parameters are removed to significantly reduce the size of a model. See Blalock et al. (2020) for a recent survey. A simple yet popular method for pruning neural networks is by global magnitude pruning Collins and Kohli (2014); Han et al. (2015): weights with lowest absolute value are set to 0. While most of the pruning techniques are performed after the model is fully trained Blalock et al. (2020), some works prune periodically during training Gale et al. (2019). However, to the best of our knowledge, pruning by magnitude is not done in a differentiable fashion. In this work, we empirically show that pruning weights with a differentiable (or differentiable almost everywhere) top-k operator in magnitude during training leads to faster convergence and better accuracy than with a "hard" one.

Top-k operator for mixture of experts. Sparse mixture of experts models (MoEs) Shazeer et al. (2017) are a class of deep learning models where only a small proportion of the model, known as experts, is activated, depending on its input. Therefore, sparse MoEs are able to increase the number of parameters without increasing the time complexity of the model. Sparse MoEs have achieved great empirical successes in computer vision Riquelme et al. (2021); Zhou et al. (2022) as well as natural language processing Shazeer et al. (2017); Lewis et al. (2021); Fedus et al. (2021). At the heart of the sparse MoE model is its routing mechanism, which determines which inputs (or tokens) are assigned to which experts. In the sparse mixture of experts literature, some works have recently proposed new top-k operators in the routing module. Hazimeh et al. (2021) proposed a binary encoding formulation to select non-zero weights. However, their formulation does not approximate the true top-k operator and sparsity is only supported at inference time, not during training. Liu et al. (2022b) proposed an optimal transport formulation supporting k-sparsity constraints and used it for sparse mixture of experts. In this work, we propose to replace the hard top-k router, which is a discontinuous function, by our smooth relaxation.

7.3 Background

Notation. We denote a permutation of $[n]$ by $\sigma = (\sigma_1, \dots, \sigma_n)$ and its inverse by σ^{-1}. When seen as a vector, we denote it $\boldsymbol{\sigma}$. We denote the set of all $n!$ permutations by Σ. Given a vector $\boldsymbol{x} \in \mathbb{R}^n$, we denote the version of $\boldsymbol{x}$ permuted according to σ by $\boldsymbol{x}_\sigma \coloneqq (x_{\sigma_1}, \dots, x_{\sigma_n})$. We denote the i-th largest value of $\boldsymbol{x} \in \mathbb{R}^n$ by $x_{[i]}$. Without loss of generality, we always sort values in descending order. The conjugate of $f(\boldsymbol{x})$ is denoted by $f^*(\boldsymbol{y}) \coloneqq \sup_{\boldsymbol{x} \in \mathbb{R}^n} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - f(\boldsymbol{x})$.

Review of operators. We denote the **argsort** operator as the permutation $\boldsymbol{\sigma}$ sorting $\boldsymbol{x} \in \mathbb{R}^n$, i.e.,

$$\operatorname{argsort}(\boldsymbol{x}) \coloneqq \boldsymbol{\sigma}, \quad \text{where} \quad x_{\sigma_1} \geq \cdots \geq x_{\sigma_n}.$$

We denote the **sort** operator as the values of $\boldsymbol{x} \in \mathbb{R}^n$ in sorted order, i.e.,

$$\operatorname{sort}(\boldsymbol{x}) \coloneqq \boldsymbol{x}_\sigma, \quad \text{where} \quad \boldsymbol{\sigma} = \operatorname{argsort}(\boldsymbol{x}).$$

The value $[\operatorname{sort}(\boldsymbol{x})]_i$ is also known as the i-th order statistic. We denote the **rank** operator as the function returning the positions of the vector $\boldsymbol{x} \in \mathbb{R}^n$ in the sorted vector. It is formally equal to the argsort's inverse permutation:

$$\operatorname{rank}(\boldsymbol{x}) \coloneqq \boldsymbol{\sigma}^{-1}, \quad \text{where} \quad \boldsymbol{\sigma} = \operatorname{argsort}(\boldsymbol{x}).$$

Smaller rank $[\operatorname{rank}(\boldsymbol{x})]_i$ means that x_i has higher value. The **top-k mask** operator returns a bit-vector encoding whether each value x_i is within the top-k values or not:

$$[\operatorname{topkmask}(\boldsymbol{x})]_i \coloneqq \begin{cases} 1, & \text{if } [\operatorname{rank}(\boldsymbol{x})]_i \leq k \\ 0, & \text{otherwise.} \end{cases}.$$

The **top-k** operator returns the values themselves if they are within the top-k values or 0 otherwise, i.e.,

$$\operatorname{topk}(\boldsymbol{x}) \coloneqq \boldsymbol{x} \circ \operatorname{topkmask}(\boldsymbol{x}),$$

where $\circ$ denotes element-wise multiplication. The **top-k in magnitude** operator is defined similarly as

$$\operatorname{topkmag}(\boldsymbol{x}) \coloneqq \boldsymbol{x} \circ \operatorname{topkmask}(|\boldsymbol{x}|).$$

To illustrate, if $x_3 \geq x_1 \geq x_2$ and $|x_2| \geq |x_3| \geq |x_1|$, then

- $\operatorname{argsort}(\boldsymbol{x}) = (3, 1, 2)$
- $\operatorname{sort}(\boldsymbol{x}) = (x_3, x_1, x_2)$
- $\operatorname{rank}(\boldsymbol{x}) = (2, 3, 1)$
- $\operatorname{topkmask}(\boldsymbol{x}) = (1, 0, 1)$
- $\operatorname{topk}(\boldsymbol{x}) = (x_1, 0, x_3)$,
- $\operatorname{topkmag}(\boldsymbol{x}) = (0, x_2, x_3)$,

where in the last three, we used $k = 2$.

Permutahedron. The permutahedron associated with a vector $\boldsymbol{w} \in \mathbb{R}^n$, a well-known object in combinatorics (Bowman, 1972; Ziegler, 2012), is the convex hull of the permutations of $\boldsymbol{w}$, i.e.,

$$P(\boldsymbol{w}) \coloneqq \operatorname{conv}(\{\boldsymbol{w}_\sigma : \sigma \in \Sigma\}) \subset \mathbb{R}^n.$$

We define the linear maximization oracles (LMO) associated with $P(\boldsymbol{w})$ by

$$\begin{aligned} f(\boldsymbol{x}, \boldsymbol{w}) &\coloneqq \max_{\boldsymbol{y} \in P(\boldsymbol{w})} \langle \boldsymbol{x}, \boldsymbol{y} \rangle \\ \boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w}) &\coloneqq \operatorname*{argmax}_{\boldsymbol{y} \in P(\boldsymbol{w})} \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \nabla_1 f(\boldsymbol{x}, \boldsymbol{w}), \end{aligned} \tag{7.1}$$

where $\nabla_1 f(\boldsymbol{x}, \boldsymbol{w})$ is technically a subgradient of f w.r.t. $\boldsymbol{x}$. The LMO can be computed in $O(n \log n)$ time. Indeed, the calculation of the LMO reduces to a sorting operation, as shown in the following known proposition. A proof is included for completeness in Appendix 7.A.1.

Proposition 7.1. *(Linear maximization oracles)*

If $w_1 \geq \cdots \geq w_n$ (if not, sort $\boldsymbol{w}$), then

$$f(\boldsymbol{x}, \boldsymbol{w}) = \sum_{i=1}^{n} w_i x_{[i]} \quad \textit{and} \quad \boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}_{\operatorname{rank}(\boldsymbol{x})}.$$

LP formulations. Let us denote the reversing permutation by $\boldsymbol{\rho} \coloneqq (n, n-1, \ldots, 1)$. Blondel et al. (2020b) showed that the sort and rank operators can be formulated as linear programs (LP) over the permutahedron:

$$\begin{aligned} \operatorname{sort}(\boldsymbol{x}) &= \boldsymbol{y}(\boldsymbol{\rho}, \boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{y} \in P(\boldsymbol{x})} \langle \boldsymbol{\rho}, \boldsymbol{y} \rangle \\ \operatorname{rank}(\boldsymbol{x}) &= \boldsymbol{y}(-\boldsymbol{x}, \boldsymbol{\rho}) = \operatorname*{argmax}_{\boldsymbol{y} \in P(\boldsymbol{\rho})} \langle -\boldsymbol{x}, \boldsymbol{y} \rangle. \end{aligned}$$

In the latter expression, the minus sign is due to the fact that we use the convention that smaller rank indicates higher value (i.e., the maximum value has rank 1).

Although not mentioned by Blondel et al. (2020b), it is also easy to express the top-k mask operator as an LP

$$\operatorname{topkmask}(\boldsymbol{x}) = \boldsymbol{y}(\boldsymbol{x}, \mathbf{1}_k) = \operatorname*{argmax}_{\boldsymbol{y} \in P(\mathbf{1}_k)} \langle \boldsymbol{x}, \boldsymbol{y} \rangle, \tag{7.2}$$

where $\mathbf{1}_k \coloneqq (\underbrace{1, \ldots, 1}_{k}, \underbrace{0, \ldots, 0}_{n-k})$. For this choice of $\boldsymbol{w}$, the permutahedron enjoys a particularly simple expression

$$P(\mathbf{1}_k) = \{\boldsymbol{y} \in \mathbb{R}^n : \langle \boldsymbol{y}, \mathbf{1} \rangle = k, \boldsymbol{y} \in [0, 1]^n\}$$

and $P(\mathbf{1}_k/k)$ is known as the capped simplex (Warmuth and Kuzmin, 2008; Blondel et al., 2020a). This is illustrated in Figure 7.2. To obtain relaxed operators, Blondel et al. (2020b) proposed to introduce regularization in (7.1) (see "recovering the previous framework" in the next section) and used a reduction to isotonic optimization.

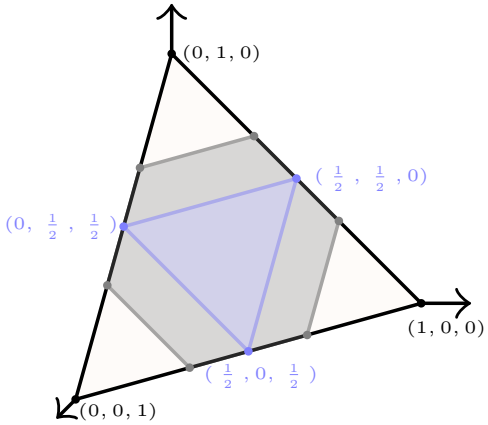

Figure 7.2: The permutahedron $P(\boldsymbol{w})$ is a polytope whose vertices are permutations of $\boldsymbol{w}$. Depending, on the choice of $\boldsymbol{w}$, it can express several known polytopes. When $\boldsymbol{w} = (1, 0, 0)$, $P(\boldsymbol{w})$ is the probability simplex (light beige), which corresponds to the top-1 setting. When $\boldsymbol{w} = (\frac{1}{2}, \frac{1}{2}, 0)$, $P(\boldsymbol{w})$ is the capped probability simplex (blue), which corresponds to the top-k setting (here, with $k = 2$). When $\boldsymbol{w} = (\frac{2}{3}, \frac{1}{3}, 0)$, $P(\boldsymbol{w})$ is an hexagon, which corresponds to the partial ranking setting (gray).

7.4 Proposed generalized framework

In this section, we generalize the framework of Blondel et al. (2020b) by adding an optional nonlinearity $\varphi(\boldsymbol{x})$. In addition to the operators covered by the previous framework, this allows us to directly express the top-k in magnitude operator, which was not possible before. We also support p-norm regularization, which allows to express differentiable *and* sparse operators when $1 < p < 2$.

Introducing a mapping φ. Consider a mapping $\varphi(\boldsymbol{x}) \coloneqq (\varphi(x_1), \ldots, \varphi(x_n))$. Given $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{w} \in \mathbb{R}^n$, we define

$$\begin{aligned} f_\varphi(\boldsymbol{x}, \boldsymbol{w}) &\coloneqq f(\varphi(\boldsymbol{x}), \boldsymbol{w}) \\ \boldsymbol{y}_\varphi(\boldsymbol{x}, \boldsymbol{w}) &\coloneqq \nabla_1 f_\varphi(\boldsymbol{x}, \boldsymbol{w}). \end{aligned}$$

When $\varphi(\boldsymbol{x}) = \boldsymbol{x}$ (identity mapping), we clearly recover the existing framework, i.e., $f_\varphi(\boldsymbol{x}, \boldsymbol{w}) = f(\boldsymbol{x}, \boldsymbol{w})$ and $\boldsymbol{y}_\varphi(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w})$.

When $\varphi(\boldsymbol{x}) \neq \boldsymbol{x}$, our framework starts to differ from the previous one, since φ affects differentiation. By the chain rule and Danskin's theorem (1966), we get

$$\begin{aligned} \boldsymbol{y}_\varphi(\boldsymbol{x}, \boldsymbol{w}) &\coloneqq \nabla_1 f_\varphi(\boldsymbol{x}, \boldsymbol{w}) \\ &= \partial\varphi(\boldsymbol{x})^\top \boldsymbol{y}(\varphi(\boldsymbol{x}), \boldsymbol{w}) \\ &= (\varphi'(x_1), \ldots, \varphi'(x_n)) \circ \boldsymbol{y}(\varphi(\boldsymbol{x}), \boldsymbol{w}), \end{aligned} \tag{7.3}$$

where $\partial\varphi(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$ denotes the Jacobian of $\varphi(\boldsymbol{x})$ and $\boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w})$ is given by Proposition 7.1.

Top-k in magnitude. As we emphasized, one advantage of our proposed generalization is that we can express the top-k in magnitude operator. Indeed, with $\varphi(x) = \frac{1}{2}x^2$, we can see from (7.2) and (7.3) that we have for all $\boldsymbol{x} \in \mathbb{R}^n$

$$\text{topkmag}(\boldsymbol{x}) = \boldsymbol{y}_\varphi(\boldsymbol{x}, \mathbf{1}_k) = \nabla_1 f_\varphi(\boldsymbol{x}, \mathbf{1}_k). \tag{7.4}$$

Obviously, for all $\boldsymbol{x} \in \mathbb{R}^n_+$, we also have $\text{topkmag}(\boldsymbol{x}) = \text{topk}(\boldsymbol{x})$. Top-k in magnitude is useful for pruning weights with small magnitude in a neural network, as we demonstrate in our experiments in §7.6.

Introducing regularization. We now explain how to make our generalized operator differentiable. We introduce convex regularization $R : \mathbb{R}^n \to \mathbb{R}$ in the dual space:

$$f^*_{\varphi,R}(\boldsymbol{y}, \boldsymbol{w}) \coloneqq f^*_\varphi(\boldsymbol{y}, \boldsymbol{w}) + R(\boldsymbol{y}),$$

where f^*_φ is the conjugate of f_φ in the first argument. Going back to the primal space, we obtain a new relaxed operator. A proof is given in Appendix 7.A.2.

Proposition 7.2. *(Relaxed operator)*

Let $R\colon \mathbb{R}^n \to \mathbb{R}$ be a convex regularizer. Then

$$\begin{aligned} f_{\varphi,R}(\boldsymbol{x}, \boldsymbol{w}) &\coloneqq \max_{\boldsymbol{y} \in \mathbb{R}^n} \langle \boldsymbol{y}, \boldsymbol{x} \rangle - f^*_\varphi(\boldsymbol{y}, \boldsymbol{w}) - R(\boldsymbol{y}) \\ &= \min_{\boldsymbol{u} \in \mathbb{R}^n} R^*(\boldsymbol{x} - \boldsymbol{u}) + f_\varphi(\boldsymbol{u}, \boldsymbol{w}) \\ \boldsymbol{y}_{\varphi,R}(\boldsymbol{x}, \boldsymbol{w}) &\coloneqq \boldsymbol{y}^\star = \nabla R^*(\boldsymbol{x} - \boldsymbol{u}^\star) = \nabla_1 f_{\varphi,R}(\boldsymbol{x}, \boldsymbol{w}). \end{aligned}$$

The mapping φ also affects conjugacy. We have the following proposition.

Proposition 7.3. *(Conjugate of f_φ in the first argument)*

If φ is convex and $\boldsymbol{w} \in \mathbb{R}^n_+$, then

$$f^*_\varphi(\boldsymbol{y}, \boldsymbol{w}) = \min_{\boldsymbol{z} \in P(\boldsymbol{w})} D_{\varphi^*}(\boldsymbol{y}, \boldsymbol{z}),$$

where $D_f(\boldsymbol{y}, \boldsymbol{z}) \coloneqq \sum_{i=1}^n z_i f(y_i/z_i)$.

A proof is given in Appendix 7.A.3. The function $(y_i, z_i) \mapsto z_i \varphi^*(y_i/z_i)$ is known as the perspective of φ^* and is jointly convex when $z_i > 0$. The function $D_f(\boldsymbol{y}, \boldsymbol{z})$ is known as the f-divergence between $\boldsymbol{y}$ and $\boldsymbol{z}$. Therefore, $f^*_\varphi(\boldsymbol{y}, \boldsymbol{w})$ can be seen as the minimum "distance" between $\boldsymbol{y}$ and $P(\boldsymbol{w})$ in the φ^*-divergence sense.

Recovering the previous framework. If $\varphi(x) = x$, then

$$\varphi^*(y_i/z_i) = \begin{cases} 0, & \text{if } y_i = z_i \\ \infty, & \text{otherwise} \end{cases}.$$

This implies that $f^*_\varphi(\boldsymbol{y}, \boldsymbol{w}) = f^*(\boldsymbol{y}, \boldsymbol{w}) = \delta_{P(\boldsymbol{w})}(\boldsymbol{y})$, the indicator function of $P(\boldsymbol{w})$, which is 0 if $\boldsymbol{y} \in P(\boldsymbol{w})$ and ∞ otherwise. In this case, we therefore obtain

$$f_{\varphi,R}(\boldsymbol{x}, \boldsymbol{w}) = \max_{\boldsymbol{y} \in P(\boldsymbol{w})} \langle \boldsymbol{y}, \boldsymbol{x} \rangle - R(\boldsymbol{y}),$$

which is exactly the relaxation of Blondel et al. (2020b).

Differentiable and sparse top-k operators. To obtain a relaxed top-k operator with our framework, we simply replace f_φ with $f_{\varphi,R}$ and $\boldsymbol{y}_\varphi$ with $\boldsymbol{y}_{\varphi,R}$ in (7.4) to define

$$\operatorname{topkmag}_R(\boldsymbol{x}) \coloneqq \boldsymbol{y}_{\varphi,R}(\boldsymbol{x}, \mathbf{1}_k) = \nabla_1 f_{\varphi,R}(\boldsymbol{x}, \mathbf{1}_k).$$

A relaxed top-k mask can be defined in a similar way, but using $\varphi(x) = x$ instead of $\varphi(x) = \frac{1}{2}x^2$. For the regularization R, we propose to use p-norms to the power p:

$$R(\boldsymbol{y}) = \frac{1}{p}\|\boldsymbol{y}\|_p^p \coloneqq \frac{1}{p}\sum_{i=1}^n |y_i|^p.$$

The choice $p = 2$ used in previous works leads to sparse outputs but is not differentiable everywhere. Any p between 1 (excluded) and 2 (excluded) leads to differentiable *and* sparse outputs. We propose to use $p = 4/3$ to obtain a differentiable everywhere operator, and $p = 2$ to obtain a differentiable a.e. operator, which is more convenient numerically. This is illustrated in Figure 7.3.

Connection with k-support and OWL norms. When $\varphi(x) = \frac{1}{2}x^2$ and $\boldsymbol{w} = \mathbf{1}_k$, we obtain

$$f_\varphi^*(\boldsymbol{y}, \boldsymbol{w}) = \frac{1}{2}\min_{\boldsymbol{z}\in[0,1]^n} \sum_{i=1}^n \frac{y_i^2}{z_i} \quad \text{s.t.} \quad \langle \boldsymbol{z}, \mathbf{1}\rangle = k,$$

which is known as the squared k-support norm Argyriou et al. (2012); McDonald et al. (2014); Eriksson et al. (2015). Our formulation is a generalization of the squared k-support norm, as it supports other choices of $\boldsymbol{w}$ and φ. For instance, we use it to define a new notion of k-support negentropy in Appendix 7.B.1. When $\varphi(\boldsymbol{x}) = |\boldsymbol{x}|$, we recover the ordered weighted lasso (OWL) norm (Zeng and Figueiredo, 2014) as

$$f_\varphi(\boldsymbol{x}, \boldsymbol{w}) = \sum_{i=1}^n w_i |x|_{[i]}.$$

With that choice of φ, it is easy to see from (7.3) that (7.4) becomes a *signed* top-k mask. Note that, interestingly, k-support and OWL norms are not defined in the same space.

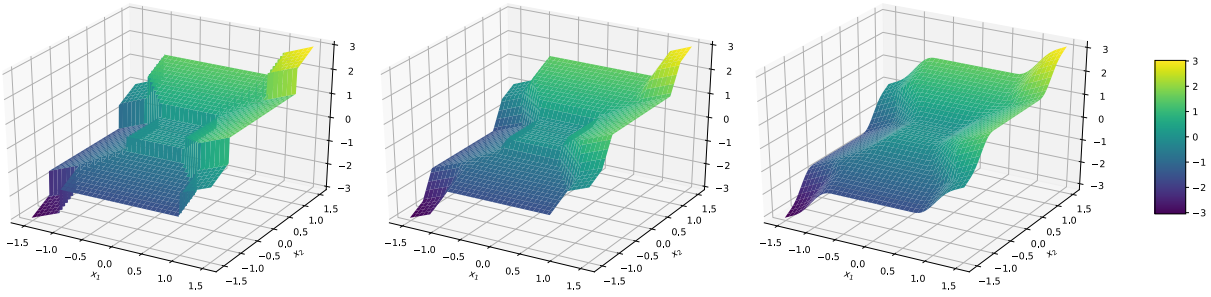

Figure 7.3: **Example of our relaxed top-k operators.** We take $\varphi(x) = \frac{1}{2}x^2$ and $k = 2$. For an input $\boldsymbol{x} = (x_1, x_2, \frac{1}{2}, 1)$, we plot $y_{\varphi,R}(\boldsymbol{x}, 1_k)_1 + y_{\varphi,R}(\boldsymbol{x}, 1_k)_2$ for $R = 0$ (left), $R = \frac{\lambda}{2}\|\boldsymbol{x}\|_2^2$ (center) and $R = \frac{\lambda}{p}\|\boldsymbol{x}\|_p^p$ with $p = \frac{4}{3}$ (right). We take $\lambda = 0.3$. While no regularization leads to a discontinuous mapping, the 2-norm regularization leads to continuity and a.e. differentiability, and the $\frac{4}{3}$-norm regularization provides a continuously differentiable mapping. We emphasize that, although in the left plot the graph looks connected, it is actually a discontinuous function. Note that our relaxed operators are **sparse** as they are exactly 0 in the center.

Biconjugate interpretation. Let us define the set of k-sparse vectors, which is nonconvex, as $S_k \coloneqq \{\boldsymbol{x} \in \mathbb{R}^n \colon \|\boldsymbol{x}\|_0 \leq k\}$, where $\|\boldsymbol{x}\|_0$ is the number of non zero elements in $\boldsymbol{x}$. We saw that if $\varphi(x) = \frac{1}{2}x^2$ then $f^*_\varphi(\boldsymbol{y}, \mathbf{1}_k)$ is the squared k-support norm. It is known to be the biconjugate (i.e., the tightest convex relaxation) of the squared L_2 norm restricted to S_k (Eriksson et al., 2015; Liu et al., 2022b). We now prove a more general result: $f^*_\varphi(\boldsymbol{y}, \mathbf{1}_k)$ is the biconjugate of $\sum_{i=1}^n \varphi^*(y_i)$ restricted to S_k.

Proposition 7.4. *(Biconjugate interpretation)*

Let $\Phi_k(\boldsymbol{y}) \coloneqq \sum_{i=1}^n \varphi^*(y_i) + \delta_{S_k}(\boldsymbol{y})$. Suppose that φ is convex. Then the biconjugate of Φ_k is given by

$$\Phi_k^{**}(\boldsymbol{y}) = f^*_\varphi(\boldsymbol{y}, \mathbf{1}_k)$$

See Appendix 7.A.4 for a proof.

7.5 Algorithms

In this section, we propose efficient algorithms for computing our operators. We first show that the calculation of our relaxed operator reduces to isotonic optimization.

Reduction to isotonic optimization. We now show how to compute $\boldsymbol{u}^\star$ in Proposition 7.2 by reduction to isotonic optimization, from which $\boldsymbol{y}^\star$ can then be recovered by $\boldsymbol{y}^\star = \nabla R^*(\boldsymbol{x} - \boldsymbol{u}^\star)$. We first recall the case $\varphi(\boldsymbol{x}) = \boldsymbol{x}$, which was already proved in existing works (Lim and Wright, 2016; Blondel et al., 2020b).

Proposition 7.5. *(Reduction, $\varphi(\boldsymbol{x}) = \boldsymbol{x}$ case)*

Suppose that $R(\boldsymbol{y}) = \sum_{i=1}^n r(y_i)$. Let σ be the permutation sorting $\boldsymbol{x}$, $\boldsymbol{s} \coloneqq \boldsymbol{x}_\sigma$ and

$$\boldsymbol{v}^\star = \operatorname*{argmin}_{v_1 \geq \cdots \geq v_n} R^*(\boldsymbol{s} - \boldsymbol{v}) + f(\boldsymbol{v}, \boldsymbol{w}).$$

Then $\boldsymbol{u}^\star$ from Proposition 7.2 is given by $\boldsymbol{u}^\star = \boldsymbol{v}^\star_{\sigma^{-1}}$.

The set $\{\boldsymbol{v} \in \mathbb{R}^n \colon v_1 \geq \cdots \geq v_n\}$ is called the monotone cone. Next, we show that a similar result is possible when $\varphi(\boldsymbol{x})$ and r^* are both even functions (sign-invariant) and increasing on $\mathbb{R}_+$.

Proposition 7.6. *(Reduction, $\varphi(\boldsymbol{x}) = \varphi(-\boldsymbol{x})$ case)*

Suppose that $R(\boldsymbol{y}) = \sum_{i=1}^n r(y_i)$ and $\varphi(\boldsymbol{x}) = (\varphi(x_1), \ldots, \varphi(x_n))$. Assume φ and r^* are both even functions (sign-invariant) and increasing on $\mathbb{R}_+$. Let σ be the permutation sorting $|\boldsymbol{x}|$, $\boldsymbol{s} \coloneqq |\boldsymbol{x}|_\sigma$ and

$$\boldsymbol{v}^\star = \operatorname*{argmin}_{v_1 \geq \cdots \geq v_n \geq 0} R^*(\boldsymbol{s} - \boldsymbol{v}) + f_\varphi(\boldsymbol{v}, \boldsymbol{w}).$$

Then, $\boldsymbol{u}^\star$ (Proposition 7.2) is equal to $\operatorname{sign}(\boldsymbol{x}) \circ \boldsymbol{v}^\star_{\sigma^{-1}}$.

See Appendix 7.A.6 for a proof. Less general results are proved in (Zeng and Figueiredo, 2014; Eriksson et al., 2015) for specific cases of φ and R. The set $\{v \in \mathbb{R}^n \colon v_1 \geq \cdots \geq v_n \geq 0\}$ is called the non-negative monotone cone. In practice, the additional non-negativity constraint is easy to handle: we can solve the isotonic optimization problem without it and truncate the solution if it is not non-negative (Németh and Németh, 2012).

Pool adjacent violator (PAV) algorithms. Under the conditions of Proposition 7.6, assuming $\boldsymbol{v}$ and $\boldsymbol{w}$ are both sorted, we have from Proposition 7.1 that

$$f(\boldsymbol{v}, \boldsymbol{w}) = \sum_{i=1}^{n} w_i v_i, \quad f_\varphi(\boldsymbol{v}, \boldsymbol{w}) = \sum_{i=1}^{n} w_i \varphi(v_i).$$

We then get that the problems in Proposition 7.5 and 7.6 are coordinate-wise separable:

$$\boldsymbol{v}^\star = \operatorname*{argmin}_{v_1 \geq \cdots \geq v_n} \sum_{i=1}^{n} h_i(v_i), \tag{7.5}$$

for $h_i(v_i) = r^*(s_i - v_i) + w_i \varphi(v_i)$. Such problems can be solved in $O(n)$ time using the pool adjacent violator (PAV) algorithm (Best et al., 2000). This algorithm works by partitioning the set $[n]$ into disjoint sets $(B_1, \cdots B_m)$, starting from $m = n$ and $B_i = \{i\}$, and by merging these sets until the isotonic condition is met. A pseudo-code is available for completness in Appendix 7.B.2. At its core, PAV simply needs a routine to solve the "pooling" subproblem

$$\gamma_B^\star = \operatorname*{argmin}_{\gamma \in \mathbb{R}} \sum_{i \in B} h_i(\gamma) \tag{7.6}$$

for any $B \subseteq [n]$. Once the optimal partition $(B_1, \cdots B_m)$ is identified, we have that

$$\boldsymbol{v}^\star = (\underbrace{\gamma_{B_1}^\star, \ldots, \gamma_{B_1}^\star}_{|B_1|}, \cdots, \underbrace{\gamma_{B_m}^\star, \ldots, \gamma_{B_m}^\star}_{|B_m|}) \in \mathbb{R}^n.$$

Because Proposition 7.5 and 7.6 require to obtain the sorting permutation σ beforehand, the total time complexity for our operators is $O(n \log n)$.

Example. Suppose $\varphi(x) = \frac{1}{2}x^2$ and $R = \frac{\lambda}{2}\|.\|_2^2$, where $\lambda > 0$. Since $R^* = \frac{1}{2\lambda}\|.\|^2$, this gives $h_i(v_i) = \frac{1}{2}(w_i v_i^2 + \frac{1}{\lambda}(s_i - v_i)^2)$. The solution of the sub-problem is then

$$\gamma_B^\star = \frac{\sum_{i \in B} s_i}{\sum_{i \in B}(\lambda w_i + 1)}.$$

Using this formula, when λ is small enough, we can upper-bound the error between the hard and relaxed operators: $\|\text{topkmag}_R(\boldsymbol{x}) - \text{topkmag}(\boldsymbol{x})\|_\infty \leq \lambda \|\boldsymbol{x}\|_\infty$. See Appendix 7.A.6 for details and for the case $p = \frac{4}{3}$.

Dykstra's alternating projection algorithm. The PAV algorithm returns an exact solution of (7.5) in $O(n)$ time. Unfortunately, it relies on element-wise dynamical array assignments, which makes it potentially slow on GPUs and TPUs. We propose an alternative to obtain faster computations. Our key insight is that by defining

$$C_1 \coloneqq \{\boldsymbol{v} \in \mathbb{R}^n \colon v_1 \geq v_2, v_3 \geq v_4, \ldots\}$$
$$C_2 \coloneqq \{\boldsymbol{v} \in \mathbb{R}^n \colon v_2 \geq v_3, v_4 \geq v_5, \ldots\},$$

one has $\{\boldsymbol{v} \in \mathbb{R}^n \colon v_1 \geq \cdots \geq v_n\} = C_1 \cap C_2$. We can therefore rewrite (7.5) as

$$\boldsymbol{v}^\star = \operatorname*{argmin}_{\boldsymbol{v} \in C_1 \cap C_2} \sum_{i=1}^{n} h_i(v_i).$$

In the case $R = \frac{1}{2}\|.\|^2$ and $\varphi(x) = x$ or $\frac{1}{2}x^2$, this reduces to a projection onto $C_1 \cap C_2$, for which we can use Dykstra's celebrated projection algorithm Boyle and Dykstra (1986); Combettes and Pesquet (2011). Jegelka et al. (2013) have used this method for computing the projection onto the intersection of submodular polytopes, whereas we project onto a single polyhedral face. When $\varphi(x) = x$, Dykstra's projection algorithm takes the simple form in Algorithm 7.7.

Algorithm 7.7. *(Dykstra's projection algorithm)*

Starting from $\boldsymbol{v}^0 = \boldsymbol{s}$, $\boldsymbol{p}^0 = \boldsymbol{q}^0 = \mathbf{0}$, *Dykstra's algorithm iterates*

$$\begin{aligned}
\boldsymbol{y}^k &= \operatorname*{argmin}_{\boldsymbol{y} \in C_1} \frac{1}{2}\|\boldsymbol{v}^k + \boldsymbol{p}^k - \boldsymbol{y}\|_2^2 + \langle \boldsymbol{y}, \boldsymbol{w} \rangle \\
\boldsymbol{p}^{k+1} &= \boldsymbol{v}^k + \boldsymbol{p}^k - \boldsymbol{y}^k \\
\boldsymbol{v}^{k+1} &= \operatorname*{argmin}_{\boldsymbol{v} \in C_2} \frac{1}{2}\|\boldsymbol{y}^k + \boldsymbol{q}^k - \boldsymbol{v}\|_2^2 + \langle \boldsymbol{v}, \boldsymbol{w} \rangle \\
\boldsymbol{q}^{k+1} &= \boldsymbol{y}^k + \boldsymbol{q}^k - \boldsymbol{v}^{k+1}.
\end{aligned}$$

Each argmin calculation corresponds to a Euclidean projection onto C_1 or C_2, which can be computed in closed form. Therefore, $\boldsymbol{v}^k$ provably converges to $\boldsymbol{v}^\star$. Each iteration of Dykstra's algorithm is learning-rate free, has linear time complexity and can be efficiently written as a matrix-vector product using a mask, which makes it particularly appealing for GPUs and TPUs. Interestingly, we find out that when $\boldsymbol{w} = \mathbf{1}_k$, then Dykstra's algorithm converges surprisingly fast to the exact solution. We validate that Dykstra leads to a faster runtime than PAV in Figure 7.4. We use 100 iterations of Dykstra, and verify that we obtain the same output as PAV.

For the general non-Euclidean R^* case, i.e., $p \neq 2$, we can use block coordinate ascent in the dual of (7.5). We can divide the dual variables into two blocks, corresponding to C_1 and C_2; see Appendix 7.A.8. In fact, it is known that in the Euclidean case, Dykstra's algorithm in the primal and block coordinate ascent in the dual are equivalent (Tibshirani, 2017). Therefore, although a vectorized implementation could be challenging for $p \neq 2$, block coordinate ascent can be seen as an elegant way to generalize Dykstra's algorithm. Convergence is guaranteed as long as each h_i is strictly convex.

Differentiation. The Jacobian of the solution of the isotonic optimization problems can be expressed in closed form for any p-norm regularization.

Proposition 7.8. *(Differentiation)*

Let $\boldsymbol{v}^\star = (\gamma^\star_{B_1}, \dots, \gamma^\star_{B_1} \cdots, \gamma^\star_{B_m}, \dots, \gamma^\star_{B_m})$ *be the optimal solution of the isotonic optimization problem with* $R = \frac{1}{p}\|.\|_p^p$ *and* $p > 0$. *Then one has that* $\boldsymbol{v}^\star$ *is differentiable with respect to* $\boldsymbol{s} = (s_1, \dots, s_n)$. *Furthermore, for any* $r \in \{1, \dots, m\}$ *and* $i \in B_r$,

$$\frac{\partial \gamma^\star_{B_r}}{\partial s_i} = \begin{cases} \frac{|\gamma^\star_{B_r} - s_i|^{q-2}}{\sum_{j \in B_r} |\gamma^\star_{B_r} - s_j|^{q-2}} & \text{if } \varphi(x) = x \\ \\ \frac{(q-1)|\gamma^\star_{B_r} - s_i|^{q-2}}{\sum_{j \in B_r} (q-1)|\gamma^\star_{B_r} - s_j|^{q-2} + w_j} & \text{if } \varphi(x) = \frac{1}{2}x^2 \end{cases}$$

where q *is such that* $\frac{1}{p} + \frac{1}{q} = 1$. *When* $i \notin B_r$, *one simply has* $\frac{\partial \gamma^\star_{B_r}}{\partial s_i} = 0$.

One then has $\partial v^\star_j / \partial s_i = \partial \gamma^\star_{B_{r_j}} / \partial s_i$ where r_j is such that $v^\star_j = \gamma^\star_{B_{r_j}}$. See Appendix 7.A.7 for a proof. Thanks to Proposition 7.8, we do not need to solve a linear system to compute the Jacobian of the solution $\boldsymbol{v}^\star$, in contrast to implicit differentiation of general optimization problems Blondel et al. (2021). In practice, this also means that we do not need to perform backpropagation through the unrolled iteratations of PAV or Dykstra's projection algorithm to obtain the gradient of a scalar loss function, in which our operator is incorporated. In particular, we do not need to store the intermediate iterates of these algorithms in memory. Along with PAV and Dykstra's

projection algorithm, we implement the corresponding Jacobian vector product routines in JAX, using Proposition 7.8.

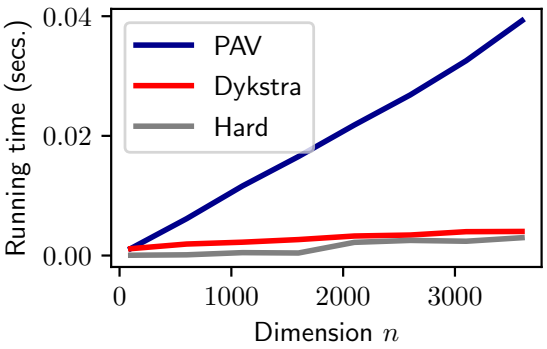

Figure 7.4: **Runtime comparison** for computing our relaxed top-k on a TPU using PAV, Dykstra, as a function of the dimension n. For each n, we set $k = \lceil n/10 \rceil$. We also compare with the hard top-k computation.

7.6 Experiments

We now demonstrate the applicability of our top-k operators through experiments. Our JAX Bradbury et al. (2018) implementation is available at the following URL. See Appendix 7.C for additional experimental details.

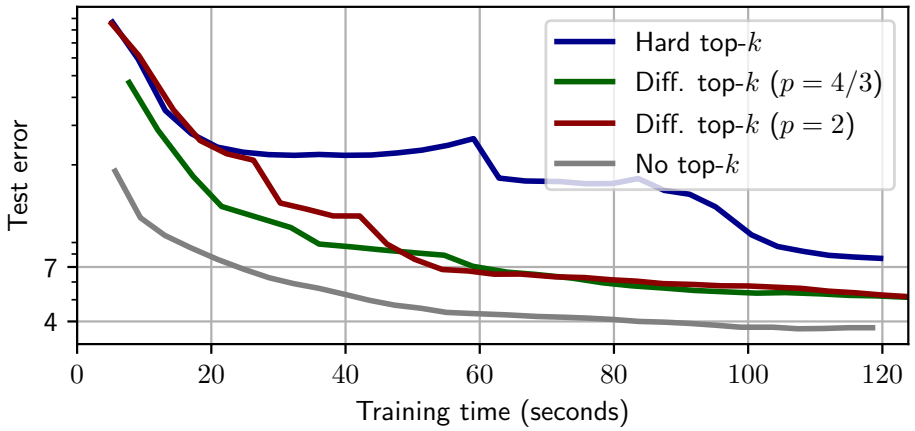

Figure 7.5: **Test error** with respect to training time when training an MLP on MNIST. We compare the baseline (grey) with the case where 90% of the weights are set to 0 by magnitude pruning, using a differentiable a.e. (red), fully differentiable (green) or a hard top-k (blue).

Weight pruning in neural networks. We experimentally validate the advantage of using a smoothed top-k for weight pruning in neural networks. We use a multilayer perceptron (MLP) with 2 hidden layers and with ReLU activation. The width of the layers are respectively 784, 32, 32 followed by a linear classification head of width 10. More precisely, our model takes as input an image $\boldsymbol{a} \in \mathbb{R}^{784}$ and outputs

$$\boldsymbol{x} = W_3\sigma(W_2\sigma(W_1\boldsymbol{a} + \boldsymbol{b}_1) + \boldsymbol{b}_2) + \boldsymbol{b}_3 \quad \text{(logits)},$$

where $W_1 \in \mathbb{R}^{32\times 784}$, $\boldsymbol{b}_1 \in \mathbb{R}^{32}$, $W_2 \in \mathbb{R}^{32\times 32}$, $\boldsymbol{b}_2 \in \mathbb{R}^{32}$, $W_1 \in \mathbb{R}^{10\times 32}$, $\boldsymbol{b}_3 \in \mathbb{R}^{10}$ and σ is a ReLU. In order to perform weight pruning we parametrize each W_i as $W_i = \text{topkmag}_R(W_i')$ and learn W_i' instead of learning W_i directly. The output is then fed into a cross-entropy loss. We compare the performance of the model when applying a hard vs differentiable top-k operator to keep only 10% of the coefficients. For the differentiable top-k, we use a regularization $R(\boldsymbol{y}) = \frac{\lambda}{p}\|\boldsymbol{y}\|^p$ with $p \in \{\frac{4}{3}, 2\}$ and $\lambda = 10^{-4}$. We find out that the model trained with the differentiable top-k trains significantly faster than the one trained with the hard top-k. We also verify that our relaxed

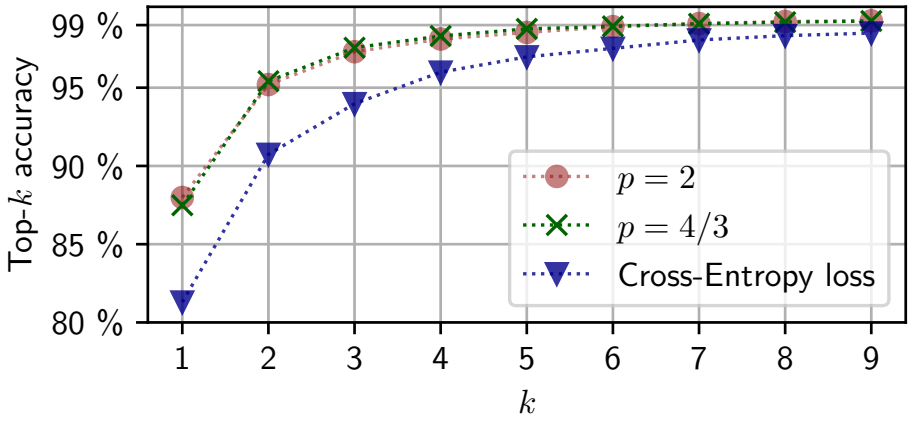

Figure 7.6: **Validation top-k accuracy** when fine-tuning a ViT-B/16 on CIFAR 100 using either the cross-entropy loss or our smooth top-3 loss for training. We have the following running times obtained with a TPUv3-8. Baseline: 9.5 sec/step, $p = 2$: 9.7 sec/step and $p = 4/3$: 10 sec/step.

top-k maintains the 10% rate of non-zero weights. Results on MNIST are displayed in Figure 7.5. We also compare with an entropy-regularized approximation of the top-k operator using the framework proposed in Cuturi et al. (2019) and adapted in Petersen et al. (2022). To guarantee the sparsity of the weights, we use the "straight-through" trick: the hard top-k is run on the forward pass but we use the gradient of the relaxed top-k in the backward pass. This method leads to a test error of 5.9%, which is comparable to the results obtained with our differentiable operators.

Smooth top-k loss. To train a neural network on a classification task, one typically minimizes the cross-entropy loss, whereas the performance of the network is evaluated using a top-k test accuracy. There is therefore a mismatch between the loss used at train time and the metric used at evaluation time. Cuturi et al. (2019) proposed to replace the cross-entropy loss with a differentiable top-k loss. In the same spirit, we propose to finetune a ViT-B/16 Dosovitskiy et al. (2020) pretrained on the ImageNet21k dataset on CIFAR 100 Krizhevsky et al. (2009) using a smooth and sparse top-k loss instead of the cross-entropy loss. We use a Fenchel-Young loss Blondel et al. (2020a). It takes as input the vector $\boldsymbol{a}$ and parameters θ of a neural network g_θ:

$$\boldsymbol{x} = g_\theta(\boldsymbol{a}) \quad \text{(logits)}$$
$$\ell(\boldsymbol{x}, \boldsymbol{t}) = f_{\varphi,R}(\boldsymbol{x}, \mathbf{1}_k) - \langle \boldsymbol{x}, \boldsymbol{t} \rangle,$$

where $f_{\varphi,R}(\boldsymbol{x}, \mathbf{1}_k)$ is given by Proposition 7.2 and $\boldsymbol{t}$ is a one-hot encoding of the class of $\boldsymbol{a}$. We set $\varphi(x) = x$ as we want a top-k mask. We consider p norm regularizations for R, where $p = 2$ or $p = 4/3$. We take $k = 3$. We use the exact same training procedure as described in Dosovitskiy et al. (2020) and use the corresponding pretrained ViT model B/16, and train our model for 100 steps. Results are reported in Figure 7.6. We find that the ViT finetuned with the smooth top-3 loss outperforms the one finetuned with the cross-entropy loss in terms of top-k error, for various k.

Sparse MoEs. Finally, we demonstrate the applicability of our proposed smooth top-k operators on a large-scale classification task using vision sparse mixture of experts (V-MoE) Riquelme et al. (2021). Vision transformers (ViTs) are made of a succession of self-attention layers and MLP layers. The idea of V-MoEs is to replace MLPs in ViTs by a sparsely-gated mixture of MLPs called experts. This way, only some of the experts are activated by a given patch token.

At the heart of the token-expert assignment lies a routing mechanism which performs a top-k operation on gate values. We focus on the MoE with expert choice routing framework Zhou et al. (2022), where each expert is assigned to k tokens. We train a S/32 variant of the V-MoE model, with 32×32 patches on the JFT-300M dataset Sun et al. (2017), a dataset with more than 305 million images. Our model has 32 experts, each assigned to $k = 28$ tokens selected among $n = 400$ at each MoE layer. We compare the validation accuracy when using the baseline (hard top-k) with our relaxed operator. We use $p = 2$ and Dykstra's projection algorithm, as we found it was the fastest method on TPU. We used the training procedure proposed by Zhou et al. (2022) to obtain a fair comparison with the baseline. Due to the large size of the JFT-300M dataset (305 million images), we performed one run, as in Liu et al. (2022b). We find that our approach improves validation performance. Results are displayed in Figure 7.7.

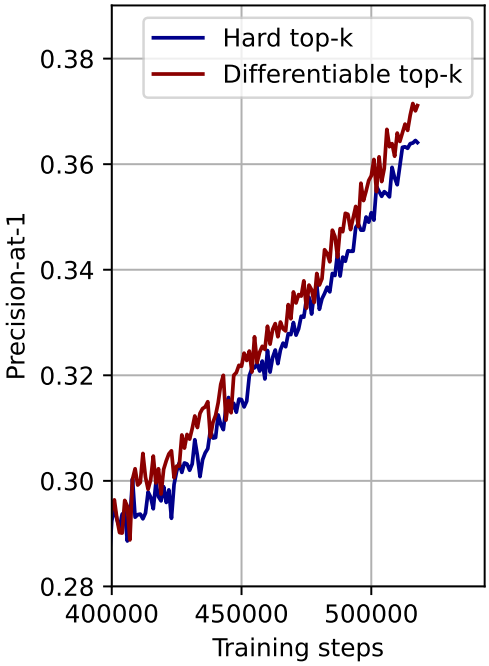

Figure 7.7: **Precision-at-1** on the JFT-300M dataset when using a hard top-k (baseline, in blue) or a differentiable a.e. one (in red) in a sparse MoE with a ViT-S/32. We zoom in on the last training steps, where our proposed method outperforms the baseline. The runtime is 10 hours for the baseline and 15 for the differentiable a.e. top-k (gradient calculation is the bottleneck here).

7.7 Discussion

Advantage of the non-linearity. As an alternative to performing a relaxed top-k operator in magnitude of $\boldsymbol{x}$, one can perform a differentiable top-k mask on $|\boldsymbol{x}|$, and then multiply the output by $\boldsymbol{x}$. This alternative would also lead to a differentiable top-k operator in magnitude. However, our operator has more principled behavior at the limit cases. For instance, as $\lambda \to \infty$, it is easy to see that the relaxed top-k mask converges to the vector $(k/n) \times 1_n$. Therefore, a rescaling by n/k is needed to obtain the identity as $\lambda \to \infty$, in contrast to our top-k in magnitude. From a theoretical point of view, the introduction of a non-linearity allows us to draw connections with the k-support norm. It also has a bi-conjugate interpretation, which we believe has an interest by itself.

Sensitivity to the choice of p. The subproblem needed within PAV enjoys a closed form only for specific choices of p. This is why we focused on $p = 2$ and $p = \frac{4}{3}$ in our experiments. However, we stress out that the proposed methods work for any choice of p. As an example, we provide the same illustration as for Figure 7.1 in Figure 7.8.

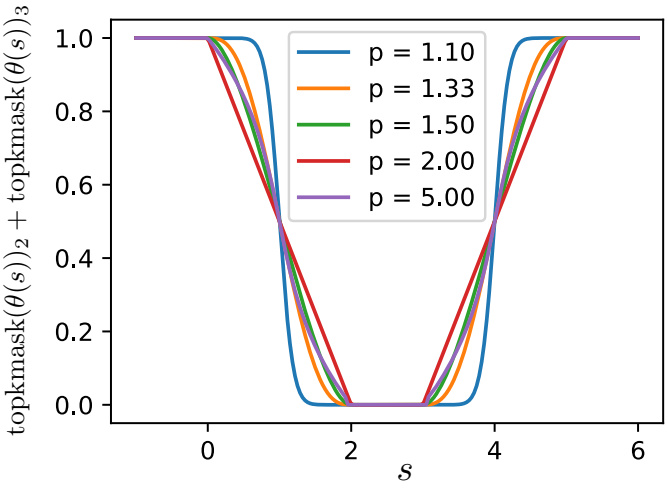

Figure 7.8: **Illustration of our differentiable and sparse top-k mask.** Same setup as for Figure 7.1, with more values for p.

7.8 Conclusion

In this work, we proposed a generalized framework to obtain fast, differentiable (or differentiable a.e.) and sparse top-k and top-k masks operators, including operators that select values in magnitude. Thanks to a reduction to isotonic optimization, we showed that these operators can be computed using either the Pool Adjacent Violators (PAV) algorithm or Dykstra's projection algorithm, the latter being faster on TPU hardware. We successfully demonstrated the usefulness of our operators for weight pruning, top-k losses and as routers in vision sparse mixture of experts.

7.A Proofs

7.A.1 Linear Maximization Oracle - Proof of Proposition 7.1

Assuming $\boldsymbol{w}$ is sorted in descending order, we have for any $\boldsymbol{x} \in \mathbb{R}^n$

$$\begin{aligned}\sum_{i=1}^{n} w_i x_{[i]} &= \max_{\sigma \in \Sigma} \langle \boldsymbol{x}_\sigma, \boldsymbol{w} \rangle \\ &= \max_{\pi \in \Sigma} \langle \boldsymbol{x}, \boldsymbol{w}_\pi \rangle \\ &= \max_{\boldsymbol{y} \in P(\boldsymbol{w})} \langle \boldsymbol{x}, \boldsymbol{y} \rangle.\end{aligned}$$

In the first line, we used that the inner product is maximized by finding the permutation σ sorting $\boldsymbol{x}$ in descending order. In the second line, we used that $\langle \boldsymbol{x}_\sigma, \boldsymbol{w} \rangle = \langle \boldsymbol{x}, \boldsymbol{w}_\pi \rangle$, if π is the inverse permutation of σ. In the third line, we used the fundamental theorem of linear programming, which guarantees that the solution happens at one of the vertices of the polytope. To summarize, if σ is the permutation sorting $\boldsymbol{x}$ in descending order, then $\boldsymbol{y}^\star = \boldsymbol{w}_{\sigma^{-1}}$.

7.A.2 Relaxed operator - Proof of Proposition 7.2

Recall that $f^*_{\varphi,R}(\boldsymbol{y}, \boldsymbol{w}) \coloneqq f^*_\varphi(\boldsymbol{y}, \boldsymbol{w}) + R(\boldsymbol{y})$. We then have

$$f_{\varphi,R}(\boldsymbol{x}, \boldsymbol{w}) = \max_{\boldsymbol{y} \in \mathbb{R}^n} \langle \boldsymbol{y}, \boldsymbol{x} \rangle - f^*_\varphi(\boldsymbol{y}, \boldsymbol{w}) - R(\boldsymbol{y}).$$

It is well-known that if h_1 and h_2 are two convex functions, then $(h_1 + h_2)^*$ is equal to the infimal convolution of h_1^* with h_2^* (Beck, 2017, Theorem 4.17):

$$(h_1 + h_2)^*(\boldsymbol{x}) = (h_1^* \square h_2^*)(\boldsymbol{x}) \coloneqq \min_{\boldsymbol{u} \in \mathbb{R}^n} h_1^*(\boldsymbol{u}) + h_2^*(\boldsymbol{x} - \boldsymbol{u}).$$

With $h_1 = f_\varphi^*$ and $h_2 = R$, we therefore get

$$f_{\varphi,R}(\boldsymbol{x}, \boldsymbol{w}) = \min_{\boldsymbol{u} \in \mathbb{R}^n} R^*(\boldsymbol{x} - \boldsymbol{u}) + f_\varphi(\boldsymbol{u}, \boldsymbol{w}).$$

Finally, the expression of $\boldsymbol{y}^\star$ follows from Danskin's theorem applied.

7.A.3 Conjugate - Proof of Proposition 7.3

We have

$$\begin{aligned} f_\varphi^*(\boldsymbol{y}, \boldsymbol{w}) &= \max_{\boldsymbol{x} \in \mathbb{R}^n} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - f_\varphi(\boldsymbol{x}, \boldsymbol{w}) \\ &= \max_{\boldsymbol{x} \in \mathbb{R}^n} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \max_{\boldsymbol{y}' \in P(\boldsymbol{w})} \langle \varphi(\boldsymbol{x}), \boldsymbol{y}' \rangle \\ &= \max_{\boldsymbol{x} \in \mathbb{R}^n} \min_{\boldsymbol{y}' \in P(\boldsymbol{w})} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \langle \varphi(\boldsymbol{x}), \boldsymbol{y}' \rangle. \end{aligned}$$

If $\boldsymbol{w} \in \mathbb{R}_+^n$, then $\boldsymbol{y}' \in \mathbb{R}_+^n$ for all $\boldsymbol{y}' \in P(\boldsymbol{w})$. Then the function $(\boldsymbol{x}, \boldsymbol{y}') \mapsto \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \langle \varphi(\boldsymbol{x}), \boldsymbol{y}' \rangle$ is concave-convex and we can switch the min and the max to obtain

$$\begin{aligned} f_\varphi^*(\boldsymbol{y}, \boldsymbol{w}) &= \min_{\boldsymbol{y}' \in P(\boldsymbol{w})} \max_{\boldsymbol{x} \in \mathbb{R}^n} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \langle \varphi(\boldsymbol{x}), \boldsymbol{y}' \rangle \\ &= \min_{\boldsymbol{y}' \in P(\boldsymbol{w})} \sum_{i=1}^n y_i' \varphi_i^*(y_i / y_i'). \end{aligned}$$

7.A.4 Biconjugate interpretation - Proof of Proposition 7.4

One has

$$\Phi_k^*(\boldsymbol{x}) = \max_{\boldsymbol{y} \in S_k} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \sum_{i=1}^n \varphi^*(y_i).$$

As in (Kyrillidis et al., 2013), let Σ_k be the set of subsets of $[n]$ with cardinality smaller than k. Then

$$\Phi_k^*(\boldsymbol{x}) = \max_{I \subset \Sigma_k} \max_{\boldsymbol{y} \in \mathbb{R}^n} \sum_{i \in I} x_i y_i - \varphi^*(y_i).$$

This gives

$$\Phi_k^*(\boldsymbol{x}) = \max_{I \subset \Sigma_k} \sum_{i \in I} \varphi(x_i) = \sum_{i=1}^k \varphi(\boldsymbol{x})_{[i]}.$$

Taking the conjugate gives the desired result.

7.A.5 Reduction to isotonic optimization

We focus on the case when φ is sign-invariant, i.e., $\varphi(\boldsymbol{x}) = \varphi(-\boldsymbol{x})$, since the case $\varphi(\boldsymbol{x}) = \boldsymbol{x}$ is already tackled in (Lim and Wright, 2016; Blondel et al., 2020b).

We first show that $\boldsymbol{u}^\star$ preserves the sign of $\boldsymbol{x}$. We do so by showing that for any $\boldsymbol{u} \in \mathbb{R}^n$, $\boldsymbol{u}' \coloneqq \operatorname{sign}(\boldsymbol{x}) \circ |\boldsymbol{u}|$ achieves smaller objective value than $\boldsymbol{u}$. Recall that

$$f_{\varphi,R}(\boldsymbol{x}, \boldsymbol{w}) = \min_{\boldsymbol{u} \in \mathbb{R}^n} R^*(\boldsymbol{x} - \boldsymbol{u}) + f_\varphi(\boldsymbol{u}, \boldsymbol{w}) = \min_{\boldsymbol{u} \in \mathbb{R}^n} R^*(\boldsymbol{x} - \boldsymbol{u}) + f(\varphi(\boldsymbol{u}), \boldsymbol{w}).$$

Clearly, we have $f(\varphi(\boldsymbol{u}'), \boldsymbol{w}) = f(\varphi(\boldsymbol{u}), \boldsymbol{w})$. Moreover, if $R(\boldsymbol{y}) = \sum_{i=1}^n r(y_i)$, then $R^*(\boldsymbol{x} - \boldsymbol{u}) = \sum_{i=1}^n r^*(x_i - u_i)$. If r^* is sign-invariant and increasing on $\mathbb{R}_+$, we then have

$$\begin{aligned} r^*(x_i - u_i') &= r^*(\operatorname{sign}(x_i)(|x_i| - |u_i|)) \\ &= r^*(|x_i| - |u_i|) \\ &\leq r^*(x_i - u_i), \end{aligned}$$

where we used the reverse triangle inequality $||x_i - |u_i|| \leq |x_i - u_i|$. We conclude that $\boldsymbol{u}^\star$ has the same sign as $\boldsymbol{x}$. From now on, we can therefore assume that $\boldsymbol{x} \in \mathbb{R}_+^n$, which implies that $\boldsymbol{u} \in \mathbb{R}_+^n$.

Since $\boldsymbol{u} \in \mathbb{R}_+^n$ and φ is increasing on $\mathbb{R}_+$, we have $u_{\sigma_1} \geq \cdots \geq u_{\sigma_n} \Rightarrow \varphi(u_{\sigma_1}) \geq \cdots \geq \varphi(u_{\sigma_n})$. We know that for all $\boldsymbol{u} \in \mathbb{R}^n$ and $\boldsymbol{w} \in \mathbb{R}^n$, we have $f(\varphi(u), \boldsymbol{w}) = \langle \varphi(\boldsymbol{u})_\sigma, \boldsymbol{w} \rangle = \langle \varphi(\boldsymbol{u}_\sigma), \boldsymbol{w} \rangle$, where σ is the permutation sorting $\boldsymbol{u}$ in descending order. From now on, let us fix σ to the permutation sorting $\boldsymbol{u}^\star$. We will show in the sequel that this is the same permutation as the one sorting $\boldsymbol{x}$. We then have

$$f_{\varphi,R}(\boldsymbol{x}, \boldsymbol{w}) = \min_{\boldsymbol{u} \in \mathbb{R}^n} R^*(\boldsymbol{x} - \boldsymbol{u}) + \langle \varphi(\boldsymbol{u}_\sigma), \boldsymbol{w} \rangle.$$

Using the change of variable $\boldsymbol{v} = \boldsymbol{u}_\sigma \Leftrightarrow \boldsymbol{v}_{\sigma^{-1}} = \boldsymbol{u}$, we obtain

$$f_{\varphi,R}(\boldsymbol{x}, \boldsymbol{w}) = \max_{v_1 \geq \cdots \geq v_n} R^*(\boldsymbol{x}_\sigma - \boldsymbol{v}) + \langle \varphi(\boldsymbol{v}), \boldsymbol{w} \rangle$$

where we used that if $R(\boldsymbol{y}) = \sum_{i=1}^n r(y_i)$, then

$$R^*(\boldsymbol{x} - \boldsymbol{u}) = R^*(\boldsymbol{x} - \boldsymbol{v}_{\sigma^{-1}}) = R^*(\boldsymbol{x}_\sigma - \boldsymbol{v}).$$

Let $\boldsymbol{s} \coloneqq \boldsymbol{x}_\sigma$. It remains to show that $s_1 \geq \cdots \geq s_n$, i.e., that $\boldsymbol{s}$ and $\boldsymbol{v}^\star$ are both in descending order. Suppose $s_j > s_i$ for some $i < j$. Let $\boldsymbol{s}'$ be a copy of $\boldsymbol{s}$ with s_i and s_j swapped. Since R^* is convex, by (Blondel et al., 2020b, Lemma 4),

$$R^*(\boldsymbol{s} - \boldsymbol{v}^\star) - R^*(\boldsymbol{s}' - \boldsymbol{v}^\star) = r^*(s_i - v_i^\star) + r^*(s_j - v_j^\star) - r^*(s_j - v_i^\star) - r^*(s_i - v_j^\star) \geq 0,$$

which contradicts the assumption that $\boldsymbol{v}^\star$ and the corresponding σ are optimal.

7.A.6 Subproblem derivation

Case $\varphi(x) = \frac{1}{2}x^2$ and $R(\boldsymbol{x}) = \frac{\lambda}{2}\|\boldsymbol{x}\|^2$. One has $h_i(\gamma) = \frac{1}{2}(w_i\gamma^2 + \frac{1}{\lambda}(s_i - \gamma)^2)$ so that $\frac{dh_i}{d\gamma} = (w_i + \frac{1}{\lambda})\gamma - \frac{1}{\lambda}s_i$. Therefore,

$$\frac{d\sum_{i \in B} h_i(\gamma)}{d\gamma} = \gamma \sum_{i \in B}(w_i + \frac{1}{\lambda}) - \frac{1}{\lambda}\sum_{i \in B} s_i.$$

Since in addition $\sum_{i \in B} h_i$ is convex we obtain that its minimum is given by canceling the derivative, hence

$$\gamma_B^\star = \frac{\sum_{i \in B} s_i}{\sum_{i \in B}(\lambda w_i + 1)}.$$

Remark. This result shows that when λ is small enough, one can control the approximation error induced by our proposed operator in comparison to the hard operator. For simplicity, let us focus on the case where there are no ties: $\forall i \neq j$, $x_i \neq x_j$. This implies that $s_1 > s_2 > \cdots > s_n$. In this case, for λ small enough, we get

$$\gamma_{\{1\}}^\star > \gamma_{\{2\}}^\star > \cdots > \gamma_{\{n\}}^\star$$

so that the optimal partition in PAV's algorithm is given by taking $B_i = \{i\}$. Therefore, $v_i^\star = \frac{s_i}{\lambda w_i + 1}$. One then has

$$\boldsymbol{u}^\star = \operatorname{sign}(\boldsymbol{x}) \circ \boldsymbol{v}^\star_{\sigma^{-1}} = \frac{\boldsymbol{x}}{\lambda \boldsymbol{w}_{\sigma^{-1}} + 1}.$$

Plugging it into $\boldsymbol{y}^\star$ given by Proposition 7.2 gives

$$\boldsymbol{y}^\star = \frac{\boldsymbol{w}_{\sigma^{-1}} \circ \boldsymbol{x}}{\lambda \boldsymbol{w}_{\sigma^{-1}} + 1}.$$

Since the hard operator is given by $\boldsymbol{w}_{\sigma^{-1}} \circ \boldsymbol{x}$, the approximation error $e_\lambda(\boldsymbol{x}, \boldsymbol{w})$ in infinite norm is then simply bounded by

$$e_\lambda(\boldsymbol{x}, \boldsymbol{w}) := \|\boldsymbol{w}_{\sigma^{-1}} \circ \boldsymbol{x}(\frac{1}{\lambda \boldsymbol{w}_{\sigma^{-1}} + 1} - 1)\|_\infty \leq \|\boldsymbol{w}\|_\infty \|\boldsymbol{x}\|_\infty \|\frac{\lambda \boldsymbol{w}}{\lambda \boldsymbol{w} + 1}\|_\infty.$$

In the topkmag case, $w = \mathbf{1}_k$, so that $e_\lambda(\boldsymbol{x}, \boldsymbol{w}) \leq \lambda \|\boldsymbol{x}\|_\infty$.

Case $\varphi(x) = x$ and $R(\boldsymbol{x}) = \frac{\lambda}{p}\|\boldsymbol{x}\|^p$ with $p = \frac{4}{3}$. One has $h_i(\gamma) = w_i\gamma + \frac{1}{4\lambda^3}(s_i - \gamma)^4$ so that $\frac{dh_i}{d\gamma} = w_i + \frac{1}{\lambda^3}(\gamma - s_i)^3$. Therefore,

$$\frac{d\sum_{i\in B} h_i(\gamma)}{d\gamma} = \frac{1}{\lambda^3}\sum_{i\in B}(\gamma - s_i)^3 + \sum_{i\in B} w_i.$$

Since in addition $\sum_{i\in B} h_i$ is convex we obtain that its minimum is given by canceling the derivative, and hence by solving the third-order polynomial equation

$$\frac{1}{\lambda^3}\sum_{i\in B}(\gamma - s_i)^3 + \sum_{i\in B} w_i = 0.$$

In practice, we solve this equation using the root solver from the *numpy* library. Note that taking $p = \frac{4}{3}$ leads to an easier subproblem than $p = \frac{3}{2}$, hence our choice for p.

Case $\varphi(x) = \frac{1}{2}x^2$ and $R(\boldsymbol{x}) = \frac{\lambda}{p}\|\boldsymbol{x}\|^p$ with $p = \frac{4}{3}$. The derivation is very similar to the previous case. Indeed, one has $h_i(\gamma) = \frac{1}{2}w_i\gamma^2 + \frac{1}{4\lambda^3}(s_i - \gamma)^4$ so that $\frac{dh_i}{d\gamma} = w_i\gamma + \frac{1}{\lambda^3}(\gamma - s_i)^3$. Therefore,

$$\frac{d\sum_{i\in B} h_i(\gamma)}{d\gamma} = \frac{1}{\lambda^3}\sum_{i\in B}(\gamma - s_i)^3 + \gamma\sum_{i\in B} w_i.$$

7.A.7 Differentiation - Proof of Proposition 7.8

Case $\varphi(x) = x$. One has $h_i(\gamma) = \frac{1}{q}|s_i - \gamma|^q + w_i v_i$. For any optimal B in PAV one has the optimality condition

$$\sum_{i\in B}(\operatorname{sign}(\gamma^\star_B - s_i)|\gamma^\star_B - s_i|^{q-1} + w_i) = 0.$$

Using the implicit function theorem gives that $\gamma^\star_B$ is differentiable, and differentiating with respect to any s_i for $i \in B$ leads to

$$\frac{\partial \gamma^\star_B}{\partial s_i} = \frac{|\gamma^\star_B - s_i|^{q-2}}{\sum_{j\in B}|\gamma^\star_B - s_j|^{q-2}}.$$

Case $\varphi(x) = \frac{1}{2}x^2$. Similar calculations lead to

$$\frac{\partial \gamma_B^\star}{\partial s_i} = \frac{(q-1)|\gamma_B^\star - s_i|^{q-2}}{\sum_{j \in B}(q-1)|\gamma_B^\star - s_j|^{q-2} + w_j}.$$

7.A.8 Dual of isotonic optimization

$$\begin{aligned}
\min_{v_1 \geq \cdots \geq v_n} \sum_{i=1}^n h_i(v_i) &= \min_{\boldsymbol{v} \in \mathbb{R}^n} \max_{\boldsymbol{\alpha} \in \mathbb{R}_+^{n-1}} \sum_{i=1}^n h_i(v_i) - \alpha_i(v_i - v_{i+1}) \\
&= \min_{\boldsymbol{v} \in \mathbb{R}^n} \max_{\boldsymbol{\alpha} \in \mathbb{R}_+^{n-1}} \sum_{i=1}^n h_i(v_i) - v_i(\alpha_i - \alpha_{i-1}) \\
&= \max_{\boldsymbol{\alpha} \in \mathbb{R}_+^{n-1}} - \left[\sum_{i=1}^n h_i^*(\alpha_i - \alpha_{i-1}) \right]
\end{aligned}$$

where $\alpha_0 \coloneqq 0$ and $\alpha_n \coloneqq 0$ are constants (i.e., not optimized). An optimal solution $\boldsymbol{v}^\star$ is recovered from $\boldsymbol{\alpha}^\star$ by $v_i^\star = (h_i^*)'(\alpha_i^\star - \alpha_{i-1}^\star)$. Since $\boldsymbol{\alpha}$ is only constrained to be non-negative, we can solve the dual by coordinate ascent. The subproblem associated with α_i, for $i \in \{1, \ldots, n-1\}$, is

$$\max_{\alpha_i \in \mathbb{R}_+} -h_i^*(\alpha_i - \alpha_{i-1}) - h_{i+1}^*(\alpha_{i+1} - \alpha_i).$$

The subproblem is a simple univariate problem with non-negative constraint. Let us define α_i' as the solution of $(h_i^*)'(\alpha_i' - \alpha_{i-1}) - (h_{i+1}^*)'(\alpha_{i+1} - \alpha_i') = 0$. The solution is then $\alpha_i^\star = [\alpha_i']_+$.

In practice, we can alternate between updating $\alpha_1, \alpha_3, \ldots$ in parallel and $\alpha_2, \alpha_4, \ldots$ in parallel. The dual variables with odd coordinates correspond to the set $C_1 = \{\boldsymbol{v} \in \mathbb{R}^n \colon v_1 \geq v_2, v_3 \geq v_4, \ldots\}$ and the dual variables with even coordinates correspond to the set $C_2 = \{\boldsymbol{v} \in \mathbb{R}^n \colon v_2 \geq v_3, v_4 \geq v_5, \ldots\}$. Coordinate ascent converges to an optimal dual solution, assuming each h_i^* is differentiable, which is equivalent to each h_i being strictly convex.

Note that the subproblem can be rewritten in primal space as

$$\begin{aligned}
&\max_{\alpha_i \in \mathbb{R}_+} -h_i^*(\alpha_i - \alpha_{i-1}) - h_{i+1}^*(\alpha_{i+1} - \alpha_i) \\
&= \max_{\alpha_i \in \mathbb{R}_+} - \left[\max_{v_i} (\alpha_i - \alpha_{i-1}) v_i - h_i(v_i) \right] - \left[\max_{v_{i+1}} (\alpha_{i+1} - \alpha_i) v_{i+1} - h_{i+1}(v_{i+1}) \right] \\
&= \min_{v_i, v_{i+1}} \alpha_{i-1} v_i + h_i(v_i) - \alpha_{i+1} v_{i+1} + h_{i+1}(v_{i+1}) + \max_{\alpha_i \in \mathbb{R}_+} \alpha_i (v_{i+1} - v_i) \\
&= \min_{v_i \geq v_{i+1}} h_i(v_i) + h_{i+1}(v_{i+1}) + \alpha_{i-1} v_i - \alpha_{i+1} v_{i+1}.
\end{aligned}$$

In fact, in the Euclidean case, it is known that Dykstra's algorithm in the primal and block coordinate ascent in the dual are equivalent (Tibshirani, 2017). Therefore, block coordinate ascent can be seen as an elegant way to generalize Dykstra's algorithm to the non-Euclidean case.

7.B Additional material

7.B.1 k-support negentropies

When $\varphi(x) = e^{x-1}$ and $\boldsymbol{w} = \mathbf{1}_k$, we obtain

$$f_\varphi^*(\boldsymbol{y}, \boldsymbol{w}) = \min_{\boldsymbol{z} \in [0,1]^n} \sum_{i=1}^n y_i \log(\frac{y_i}{z_i}) \quad \text{s.t.} \quad \langle \boldsymbol{z}, \mathbf{1} \rangle = k.$$

We call it a k-support negative entropy.

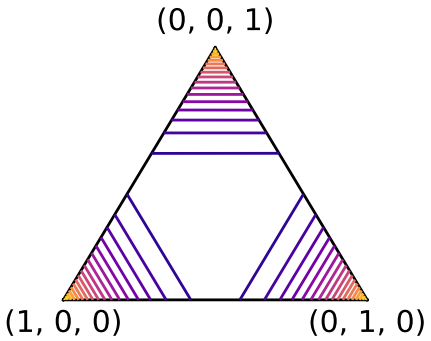

Figure 7.9: **Contours of the k-support entropy** on the simplex for $n = 3$ and $k = 2$. Lighter colors indicate lower values.

7.B.2 PAV algorithm

We present the pseudo code for PAV, adapted from Lim and Wright (2016). Recall that we define

$$\gamma_B^\star = \operatorname*{argmin}_{\gamma \in \mathbb{R}} \sum_{i \in B} h_i(\gamma).$$

Algorithm 1 Pool Adjacent Violators (PAV)

Input: Convex functions $\{h_i : \mathbb{R} \to \mathbb{R}\}_{i \in [n]}$
Initalize partitions $P \leftarrow \{\{i\} | i \in [n]\}$
Initialize $v_i \leftarrow \gamma^\star_{\{i\}}$ for all $i \in [n]$
while there exists i such that $v_i < v_{i+1}$ **do**
  Find B_{r_i} and $B_{r_{i+1}}$ in P such that $i \in B_{r_i}$ and $i+1 \in B_{r_{i+1}}$
  Remove B_{r_i} and $B_{r_{i+1}}$ from P
  Add $B_{r_i} \cup B_{r_{i+1}}$ to P
  Compute $\gamma^\star_{B_{r_i} \cup B_{r_{i+1}}}$ by solving (6)
  Assign $v_r \leftarrow \gamma^\star_{B_{r_i} \cup B_{r_{i+1}}}$ for all $r \in B_{r_i} \cup B_{r_{i+1}}$
end while
return v

7.C Experimental details

7.C.1 Weight pruning in neural networks

For our experiment on the MNIST dataset, we train the MLP using SGD with a batch size of 128 and a constant learning rate of 10^{-2}. We trained the model for 30 epochs. In terms of hardware, we use a single GPU.

7.C.2 Smooth top-k loss

For our experiment on the CIFAR-100 dataset, we train the ViT-B/16 using SGD with a momentum of 0.9 and with a batch size of 512.

For the cross-entropy loss, we follow the training procedure of Dosovitskiy et al. (2020): warmup phase until the learning rate reaches 3×10^{-3}. The model is trained for 100 steps using a cosine

learning rate scheduler. This choice of learning rate gave the best performance for this number of training steps.

For our top-3 losses: warmup phase until the learning rate reaches 5×10^{-3}. The model is trained for 100 steps using a cosine learning rate scheduler.

In terms of hardware, we use 8 TPUs.

7.C.3 Sparse MoEs

We train the V-MoE S/32 model (Riquelme et al., 2021) on the JFT-300M dataset (Sun et al., 2017). JFT is a multilabel dataset, and thus accuracy is not an appropriate metric since each image may have multiple labels. Therefore, we measure the quality of the models using the commonly-used *precision-at-1* metric (Järvelin and Kekäläinen, 2017). The training procedure is analogous to the one described in Riquelme et al. (2021), except that we replace the routing algorithm. In particular, we use the Expert Choice Routing algorithm described in Zhou et al. (2022) as our baseline, and replace the non-differentiable top-k operation used there with our differentiable approach (we perform 10 iterations of Dykstra's algorithm).

We use exactly the same hyperparameters as described in Riquelme et al. (2021), except for the fact that Expert Choice Routing does not require any auxiliary loss. Specifically, we train for 7 epochs using a batch size of 4 096. We use the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$), with a peak learning rate of 10^{-3}, warmed up for 10 000 steps and followed by linear decay. We use mild data augmentations (random cropping and horizontal flipping) and weight decay of 10^{-1} in all parameters as means of regularization. We trained both models on TPUv2-128 devices.

8 Conclusion

As time goes on, deep learning models become deeper, and datasets grow larger. In this PhD thesis, we demonstrated how this scaling facilitates the development of mathematical tools to better understand deep architectures: deep residual networks can be interpreted as neural differential equations, and Transformers as maps and flows in the space of probability measures. In addition to providing a framework for studying deep architectures, neural differential equations can be trained without the memory consumption required for storing activations, unlike standard deep networks. We first briefly recap the key contributions of the manuscript before presenting future research directions.

In Part I, we provided a mathematical framework to better understand the regime in which the infinite-depth limit of trainable residual networks corresponds to an *ordinary* differential equation (ODE). Specifically, it is sufficient for the neural network to discretize a neural ODE at initialization. We also presented a discrete adjoint method, which allows deep residual networks to be trained without memory consumption due to storing the activations in the residual layers when they are deep enough.

In Part II, we proposed using the analogy between residual-based architectures and differential equations to design and analyze new deep learning models. First, we presented Momentum ResNets, which rely on a second-order formulation of any residual network's forward rule. These can be trained with a significantly smaller memory footprint and benefit from a second-order ODE formulation. Second, we introduced Sinkformers, Transformers in which the row-wise stochastic attention matrix is replaced by a doubly stochastic attention matrix. We showed that both Transformers and Sinkformers can be interpreted as flow maps in the space of probability measures, the second corresponding to a Wasserstein gradient flow.

Finally, in Part III, we focused exclusively on Transformers. The prohibitive computational cost of scaling parameter counts in Transformers can be mitigated by routing tokens to smaller networks using Sparse Mixture of Experts with top-k gating. We proposed a sparse and differentiable variant of the top-k operator based on convex analysis and demonstrated accuracy improvements at scale. Additionally, we presented results demonstrating the capability of linear Transformers to perform autoregressive in-context learning, illustrating how a Transformer adapts its computation based on the context to estimate an internal parameter before predicting the next token.

Future Directions

Infinite Depth Limit under Realistic Assumptions. As emphasized in Chapter 3, the correspondence between deep residual networks and *ordinary* differential equations holds under the strong assumption that the network's weights are initialized as a discretization of a smooth function. This contrasts with standard random initialization, which typically leads to significantly better performance. While it is known that such random initializations result in the infinite depth limit of the network becoming a *stochastic* differential equation (SDE) before training (Marion et al., 2022), the behavior of the networks during training remains unclear. We believe that an analysis similar to that presented in Chapters 2 and 3 could provide a more realistic interpretation of deep residual networks, shedding light on their implicit bias when initialized differently from the case studied in this manuscript. In the context of Transformers, such SDEs would take the form of a McKean–Vlasov process, rather than the PDE proposed in Chapter 5.

Avoiding Memory-Computation Trade-offs. The adjoint method in neural ODEs trades-off memory usage with computational cost, and the same applies to the discrete adjoint method we proposed in Chapter 2, as well as for Momentum ResNets introduced in Chapter 4. Instead of recomputing activations during backpropagation, a potential solution could be the use of approximate activations, which would be significantly less computationally expensive to compute.

Towards More Efficient Attention. The space and time complexities of computing the attention matrix scale quadratically with respect to sequence length. Sinkformers, introduced in Chapter 5, impose a democratic principle on the attention matrix but do not promote sparsity or faster computations. However, they highlight the potential benefits of exploring alternative attention mechanisms. While Sinkhorn algorithm solves an entropic regularized optimal transport problem, sparsity could be enforced by considering squared 2-norm regularization (Blondel et al., 2018; Liu et al., 2022b).

Token Routing in Mixture of Experts. Our experimental findings in Chapter 7 demonstrate the advantages of new routing strategies in Sparse Mixture of Experts Transformers. While the use of processing unit-aware Dykstra's projection algorithm enables the scalability of our sparse and differentiable top-k operator, there is still room for improvement in the running time of our method, which we leave for future work.

Autoregressive In-Context Learning with Kernel Flows. The theoretical analysis in Chapter 6 assumes linear attention models and the commutativity of context matrices. Recent works have established connections between attention mechanisms and kernel methods (Tsai et al., 2019; Mialon et al., 2021b; Mialon, 2023; Cheng et al., 2023). In particular, Cheng et al. (2023) demonstrate that there exists a simple parameter configuration for non-linear Transformers such that they implement gradient descent in function space with respect to a Reproducing Kernel Hilbert Space (RKHS) metric. Developing a kernel-based approach for autoregressive in-context learning to account for more general mappings in the autoregressive sequence generation process would be highly valuable. This would necessitate adapting kernel ridge regression to a causal setting.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*. 16, 116

Achour, E. M., Malgouyres, F., and Gerchinovitz, S. (2021). The loss landscape of deep linear neural networks: a second-order analysis. *arXiv preprint arXiv:2107.13289*. 174, 185

Adams, R. P. and Zemel, R. S. (2011). Ranking via sinkhorn propagation. *arXiv e-prints*. 191

Agostinelli, A., Denk, T. I., Borsos, Z., Engel, J., Verzetti, M., Caillon, A., Huang, Q., Jansen, A., Roberts, A., Tagliasacchi, M., et al. (2023). Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*. 20

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, Z. (2019). Differentiable convex optimization layers. *arXiv preprint arXiv:1910.12430*. 150

Ahn, K., Cheng, X., Daneshmand, H., and Sra, S. (2023). Transformers learn to implement preconditioned gradient descent for in-context learning. *arXiv preprint arXiv:2306.00297*. 23, 168, 169, 170, 172

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. (2022). What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*. 22, 168

Allen-Zhu, Z., Li, Y., and Song, Z. (2019). A convergence theory for deep learning via over-parameterization. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 242–252. PMLR. 70

Amari, S. (1967). A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, pages 299–307. 15

Amos, B., Koltun, V., and Kolter, J. Z. (2019). The limited multi-label projection layer. *arXiv preprint arXiv:1906.08707*. 24, 35, 191

Andrica, D. and Rohan, R.-A. (2010). The image of the exponential map and some applications. In *Proc. 8th Joint Conference on Mathematics and Computer Science MaCS, Komarno, Slovakia*, pages 3–14. 122

Argyriou, A., Foygel, R., and Srebro, N. (2012). Sparse prediction with the k-support norm. *Advances in Neural Information Processing Systems*, 25. 197

Arnold, V. I. (1992). *Ordinary Differential Equations*. Springer, Berlin. 104

Arora, S., Cohen, N., and Hazan, E. (2018). On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning*, pages 244–253. PMLR. 27, 47

Arora, S., Cohen, N., Hu, W., and Luo, Y. (2019). Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32. 27, 47, 170

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*. 177

Bahdanau, D., Cho, K., and Bengio, Y. (2014a). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*. 15, 144

Bahdanau, D., Cho, K., and Bengio, Y. (2014b). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*. 168

Bai, S., Kolter, J. Z., and Koltun, V. (2019). Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32:690–701. 150

Balestriero, R., Ibrahim, M., Sobal, V., Morcos, A., Shekhar, S., Goldstein, T., Bordes, F., Bardes, A., Mialon, G., Tian, Y., Schwarzschild, A., Wilson, A. G., Geiping, J., Garrido, Q., Fernandez, P., Bar, A., Pirsiavash, H., LeCun, Y., and Goldblum, M. (2023). A cookbook of self-supervised learning. 20

Barboni, R., Peyré, G., and Vialard, F.-X. (2021). Global convergence of resnets: From finite to infinite width using linear parameterization. *arXiv preprint arXiv:2112.05531*. 26, 44, 47

Barboni, R., Peyré, G., and Vialard, F.-X. (2024). Understanding the training of infinitely deep and wide resnets with conditional optimal transport. *arXiv preprint arXiv:2403.12887*. 17

Barboni, R., Peyré, G., and Vialard, F.-X. (2022). On global convergence of ResNets: From finite to infinite width using linear parameterization. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 16385–16397. Curran Associates, Inc. 17, 70, 71

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. (2017). Spectrally-normalized margin bounds for neural networks. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, pages 6240–6249. Curran Associates, Inc. 16, 74

Bartlett, P. L., Harvey, N., Liaw, C., and Mehrabian, A. (2019). Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17. 17

Bartlett, P. L., Helmbold, D. P., and Long, P. M. (2018). Gradient descent with identity initialization efficiently learns positive definite linear transformations by deep residual networks. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 521–530. PMLR. 27, 47, 70

Bauer, F. L. (1974). Computational graphs and rounding error. *SIAM Journal on Numerical Analysis*, 11(1):87–96. 16, 116

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18. 16, 45, 116

Beck, A. (2017). *First-order methods in optimization*. SIAM. 205

Behrmann, J., Grathwohl, W., Chen, R. T., Duvenaud, D., and Jacobsen, J.-H. (2019). Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582. PMLR. 30, 116, 118, 148

Bello, I., Fedus, W., Du, X., Cubuk, E. D., Srinivas, A., Lin, T.-Y., Shlens, J., and Zoph, B. (2021). Revisiting resnets: Improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 34. 25, 42

Berrada, L., Zisserman, A., and Kumar, M. P. (2018). Smooth loss functions for deep top-k classification. *International Conference on Learning Representations*. 191

Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J.-P., and Bach, F. (2020). Learning with differentiable pertubed optimizers. *Advances in neural information processing systems*, 33:9508–9519. 24, 192

Best, M. J., Chakravarti, N., and Ubhaya, V. A. (2000). Minimizing separable convex functions subject to simple chain constraints. *SIAM Journal on Optimization*, 10(3):658–672. 36, 199

Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.". 176

Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., and Guttag, J. (2020). What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146. 192

Blondel, M. (2019). Structured prediction with projection oracles. *Advances in neural information processing systems*, 32. 191

Blondel, M., Berthet, Q., Cuturi, M., Frostig, R., Hoyer, S., Llinares-López, F., Pedregosa, F., and Vert, J.-P. (2021). Efficient and modular implicit differentiation. *arXiv preprint arXiv:2105.15183*. 148, 200

Blondel, M., Martins, A. F., and Niculae, V. (2020a). Learning with fenchel-young losses. *J. Mach. Learn. Res.*, 21(35):1–69. 194, 202

Blondel, M. and Roulet, V. (2024). The elements of differentiable programming. *arXiv preprint arXiv:2403.14606*. 24

Blondel, M., Seguy, V., and Rolet, A. (2018). Smooth and sparse optimal transport. In *International conference on artificial intelligence and statistics*, pages 880–889. PMLR. 212

Blondel, M., Teboul, O., Berthet, Q., and Djolonga, J. (2020b). Fast differentiable sorting and ranking. In *International Conference on Machine Learning*, pages 950–959. PMLR. 24, 35, 191, 192, 194, 195, 196, 198, 205, 206

Borsos, Z., Marinier, R., Vincent, D., Kharitonov, E., Pietquin, O., Sharifi, M., Roblek, D., Teboul, O., Grangier, D., Tagliasacchi, M., et al. (2023). Audiolm: a language modeling approach to audio generation. *IEEE/ACM transactions on audio, speech, and language processing*, 31:2523–2533. 20

Boursier, E., Pillaud-Vivien, L., and Flammarion, N. (2022). Gradient flow dynamics of shallow ReLU networks for square loss and orthogonal inputs. In Koyejo, S., Mohamed, S., Agarwal,

A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 20105–20118. Curran Associates, Inc. 70

Bowman, V. (1972). Permutation polyhedra. *SIAM Journal on Applied Mathematics*, 22(4):580–589. 194

Boyle, J. P. and Dykstra, R. L. (1986). A method for finding projections onto the intersection of convex sets in hilbert spaces. In *Advances in order restricted statistical inference*, pages 28–47. Springer. 36, 199

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., and Wanderman-Milne, S. (2018). Jax: composable transformations of python+ numpy programs. *URL http://github. com/google/jax*. 176, 201

Brezis, H. and Brézis, H. (2011). *Functional analysis, Sobolev spaces and partial differential equations*, volume 2. Springer. 55

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*. 15, 19, 22, 31, 144, 146, 166, 168

Castin, V., Ablin, P., and Peyré, G. (2024). How smooth is attention? In *ICML 2024*. 22

Cettolo, M., Niehues, J., Stüker, S., Bentivogli, L., and Federico, M. (2014). Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*, volume 57. 33, 154

Chang, B., Meng, L., Haber, E., Ruthotto, L., Begert, D., and Holtham, E. (2018). Reversible architectures for arbitrarily deep residual neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32. 30, 116, 119

Chapelle, O. and Wu, M. (2010). Gradient descent optimization of smoothed information retrieval metrics. *Information retrieval*, 13(3):216–235. 191

Charlier, B., Feydy, J., Glaunès, J., Collin, F.-D., and Durif, G. (2021). Kernel operations on the gpu, with autodiff, without memory overflows. *Journal of Machine Learning Research*, 22(74):1–6. 147

Chen, R. T. Q. (2018). torchdiffeq. 17

Chen, T., Zhang, Z., Cheng, Y., Awadallah, A., and Wang, Z. (2022). The principle of diversity: Training stronger vision transformers calls for reducing all levels of redundancy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12020–12030. 173

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583. 17, 18, 26, 31, 42, 44, 45, 68, 69, 71, 116, 118, 120, 127, 140, 146, 149

Cheng, X., Chen, Y., and Sra, S. (2023). Transformers implement functional gradient descent to learn non-linear functions in context. *arXiv preprint arXiv:2312.06528*. 212

Chizat, L. and Bach, F. (2020). Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on learning theory*, pages 1305–1338. PMLR. 16, 22

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2023). Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113. 23, 166

Chun, I. Y., Huang, Z., Lim, H., and Fessler, J. (2020). Momentum-net: Fast and convergent iterative neural network for inverse problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 117

Cohen, A.-S., Cont, R., Rossier, A., and Xu, R. (2021). Scaling properties of deep residual networks. In *International Conference on Machine Learning*, pages 2039–2048. PMLR. 26, 44, 47, 70, 79

Collins, M. D. and Kohli, P. (2014). Memory bounded deep convolutional networks. *arXiv preprint arXiv:1412.1442*. 192

Combettes, P. L. and Pesquet, J.-C. (2011). Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer. 36, 199

Cont, R., Rossier, A., and Xu, R. (2022). Convergence and implicit regularization properties of gradient descent for deep residual networks. *arXiv preprint arXiv:2204.07261*. 26, 44, 70, 79

Conway, J. B. (2012). *Functions of one complex variable II*, volume 159. Springer Science & Business Media. 131

Cordonnier, J.-B., Mahendran, A., Dosovitskiy, A., Weissenborn, D., Uszkoreit, J., and Unterthiner, T. (2021). Differentiable patch selection for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2351–2360. 192

Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., and Ho, S. (2019). Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*. 44

Craven, T. and Csordas, G. (2002). Iterated laguerre and turán inequalities. *J. Inequal. Pure Appl. Math*, 3(3):14. 133

Cuchiero, C., Larsson, M., and Teichmann, J. (2020). Deep neural networks, generic universal interpolation, and controlled odes. *SIAM Journal on Mathematics of Data Science*, 2(3):901–919. 44

Culver, W. J. (1966). On the existence and uniqueness of the real logarithm of a matrix. *Proceedings of the American Mathematical Society*, 17(5):1146–1151. 122, 133

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300. 21, 31, 32, 144, 147, 192

Cuturi, M., Teboul, O., Niles-Weed, J., and Vert, J.-P. (2020). Supervised quantile normalization for low rank matrix factorization. In *International Conference on Machine Learning*, pages 2269–2279. PMLR. 148

Cuturi, M., Teboul, O., and Vert, J.-P. (2019). Differentiable ranking and sorting using optimal transport. *Advances in neural information processing systems*, 32. 24, 191, 202

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314. 31, 117

Danskin, J. M. (1966). The theory of max-min, with applications. *SIAM Journal on Applied Mathematics*, 14(4):641–664. 195

Daubechies, I., Defrise, M., and De Mol, C. (2004). An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457. 127

De Bie, G., Peyré, G., and Cuturi, M. (2019). Stochastic deep networks. In *International Conference on Machine Learning*, pages 1556–1565. PMLR. 21, 146

Debnath, L. and Bhatta, D. (2014). *Integral Transforms and Their Applications*. CRC press, Boca Raton, third edition. 98

Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A. P., Caron, M., Geirhos, R., Alabdulmohsin, I., et al. (2023). Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pages 7480–7512. PMLR. 23

Demailly, J.-P. (2016). *Analyse numérique et équations différentielles-4ème Ed*. EDP sciences. 54

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009a). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. 14, 17

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009b). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee. 31, 119, 124

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. 19, 166

DeVore, R., Hanin, B., and Petrova, G. (2021). Neural network approximation. *Acta Numerica*, 30:327–444. 16

Dong, C., Liu, L., Li, Z., and Shang, J. (2020). Towards adaptive residual network training: A neural-ODE perspective. In Daumé III, H. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 2616–2626. PMLR. 18, 25, 68

Dong, Y., Cordonnier, J.-B., and Loukas, A. (2021). Attention is not all you need: Pure attention loses rank doubly exponentially with depth. *arXiv preprint arXiv:2103.03404*. 150

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. 19, 20, 25, 31, 42, 144, 145, 155, 173, 202, 209

Dragomir, S. S. (2003). *Some Gronwall Type Inequalities and Applications*. Nova Science Publishers. 83

Dryanov, D. and Rahman, Q. (1999). Approximation by entire functions belonging to the laguerre–polya class. *Methods and Applications of Analysis*, 6(1):21–38. 133

Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. (2019). Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685. PMLR. 26, 44

E, W., Han, J., and Li, Q. (2019). A mean-field optimal control formulation of deep learning. *Research in the Mathematical Sciences*, 6:10. 18, 25, 68

Eriksson, A., Thanh Pham, T., Chin, T.-J., and Reid, I. (2015). The k-support norm and convex envelopes of cardinality and rank. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3349–3357. 197, 198

Fedus, W., Dean, J., and Zoph, B. (2022). A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*. 35, 190

Fedus, W., Zoph, B., and Shazeer, N. (2021). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. 192

Filippov, A. F. (1988). *Differential Equations with Discontinuous Righthand Sides*. Springer, Dordrecht. 105

Fournier, Q., Caron, G. M., and Aloise, D. (2023). A practical survey on faster and lighter transformers. *ACM Computing Surveys*, 55(14s):1–40. 20, 168

Frankle, J. and Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*. 190

Frei, S., Cao, Y., and Gu, Q. (2019). Algorithm-dependent generalization bounds for overparameterized deep residual networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32, pages 14769–14779. Curran Associates, Inc. 70

Furuya, T., de Hoop, M. V., and Peyré, G. (2024). Transformers are universal in-context learners. *arXiv preprint arXiv:2408.01367*. 22

Gale, T., Elsen, E., and Hooker, S. (2019). The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*. 192

Gantmacher, F. R. (1959). The theory of matrices. *Chelsea, New York*, Vol. I. 123, 133, 134

Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. (2022). What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598. 22, 168

Geshkovski, B., Letrouit, C., Polyanskiy, Y., and Rigollet, P. (2023). The emergence of clusters in self-attention dynamics. *arXiv:2305.05465*. 22, 29, 76

Gholami, A., Keutzer, K., and Biros, G. (2019). Anode: Unconditionally accurate memory-efficient gradients for neural odes. *arXiv preprint arXiv:1902.10298*. 18, 116

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pages 249–256. PMLR. 68

Goel, S., Gollakota, A., Jin, Z., Karmalkar, S., and Klivans, A. (2020). Superpolynomial lower bounds for learning one-layer neural networks using gradient descent. In Daumé III, H. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 3587–3596. PMLR. 103

Gomez, A. N., Ren, M., Urtasun, R., and Grosse, R. B. (2017). The reversible residual network: Backpropagation without storing activations. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc. 27, 28, 30, 42, 45, 69, 116, 119, 125, 127

Goodfellow, I., Bengio, Y., and Courville, A. (2016a). *Deep Learning*. MIT Press. 71, 73

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016b). *Deep learning*, volume 1. MIT press Cambridge. 114

Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R. S., and Guo, E. (2016). On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*. 150

Gould, S., Hartley, R., and Campbell, D. (2019). Deep declarative networks: A new hope. *arXiv preprint arXiv:1909.04866*. 150

Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I., and Duvenaud, D. (2018). Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*. 44

Gregor, K. and LeCun, Y. (2010). Learning fast approximations of sparse coding. In *Proceedings of the 27th international conferenceon machine learning*, pages 399–406. 127

Greydanus, S., Dzamba, M., and Yosinski, J. (2019). Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, pages 15353–15363. 18, 44

Griewank, A. and Walther, A. (2008a). *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM. 116

Griewank, A. and Walther, A. (2008b). *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM. 148

Grover, A., Wang, E., Zweig, A., and Ermon, S. (2019). Stochastic optimization of sorting networks via continuous relaxations. In *International Conference on Learning Representations*. 24, 192

Guo, M.-H., Cai, J.-X., Liu, Z.-N., Mu, T.-J., Martin, R. R., and Hu, S.-M. (2021). Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199. 33, 152, 153

Gusak, J., Markeeva, L., Daulbaev, T., Katrutsa, A., Cichocki, A., and Oseledets, I. (2020). Towards understanding normalization in neural odes. *arXiv preprint arXiv:2004.09222*. 17, 116

Haber, E. and Ruthotto, L. (2017a). Stable architectures for deep neural networks. *Inverse Problems*, 34:014004. 18, 25, 68

Haber, E. and Ruthotto, L. (2017b). Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004. 116

Hairer, E., Lubich, C., and Wanner, G. (2006). *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media. 119, 121

Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28. 190, 192

Hanson, J. and Raginsky, M. (2022). Fitting an immersed submanifold to data via Sussmann's orbit theorem. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 5323–5328. 17, 69

Hartfiel, D. J. (1995). Dense sets of diagonalizable matrices. *Proceedings of the American Mathematical Society*, 123(6):1669–1672. 122

Hayou, S. (2023). On the infinite-depth limit of finite-width neural networks. *Transactions on Machine Learning Research*. 70

Hayou, S., Clerico, E., He, B., Deligiannidis, G., Doucet, A., and Rousseau, J. (2021). Stable resnet. In *International Conference on Artificial Intelligence and Statistics*, pages 1324–1332. PMLR. 25

Hazimeh, H., Zhao, Z., Chowdhery, A., Sathiamoorthy, M., Chen, Y., Mazumder, R., Hong, L., and Chi, E. (2021). Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *Advances in Neural Information Processing Systems*, 34:29335–29347. 192

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738. 117

He, K. and Sun, J. (2015). Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5353–5360. 14

He, K., Zhang, X., Ren, S., and Sun, J. (2015a). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034. 47

He, K., Zhang, X., Ren, S., and Sun, J. (2015b). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1026–1034. IEEE. 68

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 14, 15, 19, 25, 42, 51, 52, 66, 68, 109, 114, 118, 119, 124, 146, 168

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. 14, 15, 25, 42, 52, 66, 71

Hein, M., Audibert, J.-Y., and Luxburg, U. v. (2007). Graph laplacians and their convergence on random neighborhood graphs. *Journal of Machine Learning Research*, 8(6). 146

Hendrycks, D. and Gimpel, K. (2016). Gaussian Error Linear Units (GELUs). *arXiv:1606.08415*. 76

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. (2022). Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*. 166

Hyvärinen, A. and Dayan, P. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4). 13

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR. 66

Jacobsen, J.-H., Smeulders, A. W., and Oyallon, E. (2018). i-revnet: Deep invertible networks. In *International Conference on Learning Representations*. 16, 45, 116, 119

Järvelin, K. and Kekäläinen, J. (2017). Ir evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, volume 51, pages 243–250. ACM New York, NY, USA. 210

Jegelka, S., Bach, F., and Sra, S. (2013). Reflection methods for user-friendly submodular optimization. *Advances in Neural Information Processing Systems*, 26. 199

Jelassi, S., Sander, M., and Li, Y. (2022). Vision transformers provably learn spatial structure. *Advances in Neural Information Processing Systems*, 35:37822–37836. 173

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. (2023). Mistral 7b. *arXiv preprint arXiv:2310.06825*. 20, 166, 168

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., et al. (2024). Mixtral of experts. *arXiv preprint arXiv:2401.04088*. 23

Jordan, R., Kinderlehrer, D., and Otto, F. (1998). The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17. 149

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Zidek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589. 20

Jurafsky, D. and Martin, J. H. (2009). *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall. 167

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. (2020). Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR. 20, 168

Kato, T. (2013). *Perturbation theory for linear operators*, volume 132. Springer Science & Business Media. 187

Kazemnejad, A., Padhi, I., Ramamurthy, K. N., Das, P., and Reddy, S. (2023). The impact of positional encoding on length generalization in transformers. *arXiv preprint arXiv:2305.19466*. 168

Kidger, P. (2022). On neural differential equations. *arXiv preprint arXiv:2202.02435*. 17, 18, 25, 42, 44, 69

Kim, H., Papamakarios, G., and Mnih, A. (2021). The lipschitz constant of self-attention. In *International Conference on Machine Learning*, pages 5562–5571. PMLR. 21, 29, 76, 148, 150, 151

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 13, 160, 177

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. (2019). Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*, 6(2):8. 114

Krantz, S. G. and Parks, H. R. (2012). *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media. 148

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto. 78

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. In *Toronto, ON, Canada*. 202

Krizhevsky, A., Nair, V., and Hinton, G. (2010). Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*, 5. 31, 124

Kyrillidis, A., Becker, S., Cevher, V., and Koch, C. (2013). Sparse projections onto the simplex. In *International Conference on Machine Learning*, pages 235–243. PMLR. 205

Lapin, M., Hein, M., and Schiele, B. (2015). Top-k multiclass svm. *Advances in Neural Information Processing Systems*, 28. 191

Lapin, M., Hein, M., and Schiele, B. (2016). Loss functions for top-k error: Analysis and insights. In *Proc. of CVPR*. 191

Laurent, B. and Massart, P. (2000). Adaptive estimation of a quadratic functional by model selection. *The Annals of Statistics*, 28:1302–1338. 106

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444. 114

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. 15

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In Montavon, G., Orr, G., and Müller, K.-R., editors, *Neural Networks: Tricks of the Trade: Second Edition*, pages 9–48, Berlin. Springer. 68

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. (2019a). Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR. 31, 144, 145, 152

Lee, M., Lee, J., Jang, H. J., Kim, B., Chang, W., and Hwang, K. (2019b). Orthogonality constrained multi-head attention for keyword spotting. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 86–92. IEEE. 173

Levin, B. Y. (1996). *Lectures on entire functions*, volume 150. American Mathematical Soc. 131

Lewis, M., Bhosale, S., Dettmers, T., Goyal, N., and Zettlemoyer, L. (2021). Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pages 6265–6274. PMLR. 192

Li, H., Yang, Y., Chen, D., and Lin, Z. (2018). Optimization algorithm inspired deep neural network structure design. In *Asian Conference on Machine Learning*, pages 614–629. PMLR. 117

Li, Q., Lin, T., and Shen, Z. (2019). Deep learning via dynamical systems: An approximation perspective. *arXiv preprint arXiv:1912.10382*. 31, 42, 44, 117

Li, Y., Ildiz, M. E., Papailiopoulos, D., and Oymak, S. (2023). Transformers as algorithms: Generalization and stability in in-context learning. In *International Conference on Machine Learning*, pages 19565–19594. PMLR. 22, 168

Li, Z., Luo, Y., and Lyu, K. (2021). Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning. In *International Conference on Learning Representations*. 70

Lim, C. H. and Wright, S. J. (2016). Efficient bregman projections onto the permutahedron and related polytopes. In *Artificial Intelligence and Statistics*, pages 1205–1213. PMLR. 198, 205, 209

Liu, C., Zhu, L., and Belkin, M. (2020). On the linearity of large non-linear models: when and why the tangent kernel is constant. *Advances in Neural Information Processing Systems*, 33. 26, 44

Liu, C., Zhu, L., and Belkin, M. (2022a). Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59:85–116. 69, 70

Liu, T., Blondel, M., Riquelme, C., and Puigcerver, J. (2024). Routers in vision mixture of experts: An empirical study. *arXiv preprint arXiv:2401.15969*. 24

Liu, T., Puigcerver, J., and Blondel, M. (2022b). Sparsity-constrained optimal transport. *arXiv preprint arXiv:2209.15466*. 192, 198, 203, 212

Loyka, S. (2015). On singular value inequalities for the sum of two matrices. *arXiv:1507.06630*. 106

Lu, Y., Li, Z., He, D., Sun, Z., Dong, B., Qin, T., Wang, L., and Liu, T.-Y. (2019). Understanding and improving transformer from a multi-particle dynamic system point of view. *arXiv preprint arXiv:1906.02762*. 22, 29, 44, 76

Lu, Y., Ma, C., Lu, Y., Lu, J., and Ying, L. (2020). A mean field analysis of deep resnet and beyond: Towards provably optimization via overparameterization from depth. In *International Conference on Machine Learning*, pages 6426–6436. PMLR. 26, 44

Lu, Y., Zhong, A., Li, Q., and Dong, B. (2018). Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *International Conference on Machine Learning*, pages 3276–3285. PMLR. 44, 116, 117, 146

Luise, G., Rudi, A., Pontil, M., and Ciliberto, C. (2018). Differential properties of sinkhorn approximation for learning with wasserstein distance. *arXiv preprint arXiv:1805.11897*. 148

Luk, J. (2017). Notes on existence and uniqueness theorems for ODEs. URL: `http://web.stanford.edu/~jluk/math63CMspring17/Existence.170408.pdf`. 104

Lyu, K. and Li, J. (2020). Gradient descent maximizes the margin of homogeneous neural networks. In *International Conference on Learning Representations*. 70

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics. 33, 154

MacDonald, L. E., Saratchandran, H., Valmadre, J., and Lucey, S. (2022). A global analysis of global optimisation. *arXiv:2210.05371*. 70

Maclaurin, D., Duvenaud, D., and Adams, R. (2015). Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pages 2113–2122. PMLR. 118, 120, 128, 139

Mahankali, A., Hashimoto, T. B., and Ma, T. (2023). One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. *arXiv preprint arXiv:2307.03576*. 22, 168, 169, 170, 172

Malaviya, C., Ferreira, P., and Martins, A. F. (2018). Sparse and constrained attention for neural machine translation. *arXiv preprint arXiv:1805.08241*. 191

Maleki, M., Habiba, M., and Pearlmutter, B. A. (2021). Heunnet: Extending resnet using heun's method. In *2021 32nd Irish Signals and Systems Conference (ISSC)*, pages 1–6. IEEE. 50

Marion, P. (2023). Generalization bounds for neural ordinary differential equations and deep residual networks. *arXiv:2305.06648*. 17, 69

Marion, P., Fermanian, A., Biau, G., and Vert, J.-P. (2022). Scaling resnets in the large-depth regime. *arXiv preprint arXiv:2206.06929*. 26, 44, 47, 68, 70, 76, 79, 212

Marion, P., Wu, Y.-H., Sander, M. E., and Biau, G. (2024). Implicit regularization of deep residual networks towards neural ODEs. In *The Twelfth International Conference on Learning Representations*. 5

Marshall, N. F. and Coifman, R. R. (2019). Manifold learning with bi-stochastic kernels. *IMA Journal of Applied Mathematics*, 84(3):455–482. 146, 151, 159

Martens, J. and Sutskever, I. (2012). Training deep and recurrent networks with hessian-free optimization. In *Neural networks: Tricks of the trade*, pages 479–535. Springer. 116

Martins, A. F. and Kreutzer, J. (2017). Learning what's easy: Fully differentiable neural easy-first taggers. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 349–362. 191

Massaroli, S., Poli, M., Park, J., Yamashita, A., and Asama, H. (2020). Dissecting neural odes. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3952–3963. Curran Associates, Inc. 18, 25, 68, 117

McDonald, A. M., Pontil, M., and Stamos, D. (2014). Spectral k-support norm regularization. *Advances in neural information processing systems*, 27. 197

Mialon, G. (2023). On inductive biases for machine learning in data constrained settings. *arXiv preprint arXiv:2302.10692*. 212

Mialon, G., Chen, D., d'Aspremont, A., and Mairal, J. (2021a). A trainable optimal transport embedding for feature aggregation and its relationship to attention. In *ICLR 2021-The Ninth International Conference on Learning Representations*. 146

Mialon, G., Chen, D., Selosse, M., and Mairal, J. (2021b). Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*. 212

Michel, P., Levy, O., and Neubig, G. (2019). Are sixteen heads really better than one? *Advances in neural information processing systems*, 32. 19, 168, 173

Milanfar, P. (2013). Symmetrizing smoothing filters. *SIAM Journal on Imaging Sciences*, 6(1):263–284. 146

Nath, S., Khadilkar, H., and Bhattacharyya, P. (2024). Transformers are expressive, but are they expressive enough for regression? *arXiv preprint arXiv:2402.15478*. 22

Németh, A. and Németh, S. (2012). How to project onto the monotone nonnegative cone using pool adjacent violators type algorithms. *arXiv preprint arXiv:1201.2343*. 198

Neyshabur, B., Tomioka, R., and Srebro, N. (2014). In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv:1412.6614*. 26, 69, 70

Nguegnang, G. M., Rauhut, H., and Terstiege, U. (2021). Convergence of gradient descent for learning linear neural networks. *arXiv preprint arXiv:2108.02040*. 173, 174, 185

Nguyen, Q. N. and Mondelli, M. (2020). Global convergence of deep networks with one wide layer followed by pyramidal topology. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11961–11972. Curran Associates, Inc. 70, 98, 103, 107

Nguyen, T. M., Baraniuk, R. G., Bertozzi, A. L., Osher, S. J., and Wang, B. (2020). Momentum-rnn: Integrating momentum into recurrent neural networks. *arXiv preprint arXiv:2006.06919*. 117

Nichani, E., Damian, A., and Lee, J. D. (2024). How transformers learn causal structure with gradient descent. *arXiv preprint arXiv:2402.14735*. 23

Niculae, V., Martins, A., Blondel, M., and Cardie, C. (2018). Sparsemap: Differentiable sparse structured inference. In *International Conference on Machine Learning*, pages 3799–3808. PMLR. 146

Niepert, M., Minervini, P., and Franceschi, L. (2021). Implicit mle: backpropagating through discrete exponential family distributions. *Advances in Neural Information Processing Systems*, 34:14567–14579. 191

Norcliffe, A., Bodnar, C., Day, B., Simidjievski, N., and Liò, P. (2020). On second order behaviour in augmented neural odes. *arXiv preprint arXiv:2006.07220*. 117

Ott, K., Katiyar, P., Hennig, P., and Tiemann, M. (2021). Resnet after all: Neural {ode}s and their numerical solution. In *International Conference on Learning Representations*. 44

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (2019). fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*. 154

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318. 154

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. 16, 51, 65, 116, 139, 140, 152, 176

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8024–8035. Curran Associates, Inc. 109

Peng, C., Zhang, X., Yu, G., Luo, G., and Sun, J. (2017). Large kernel matters–improve semantic segmentation by global convolutional network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4353–4361. 27, 42, 44, 114, 150

Perko, L. (2013). *Differential equations and dynamical systems*, volume 7. Springer Science & Business Media. 135

Pesme, S., Pillaud-Vivien, L., and Flammarion, N. (2021). Implicit bias of sgd for diagonal linear networks: a provable benefit of stochasticity. *Advances in Neural Information Processing Systems*, 34:29218–29230. 171

Petersen, F., Borgelt, C., Kuehne, H., and Deussen, O. (2021). Differentiable sorting networks for scalable sorting and ranking supervision. In *International Conference on Machine Learning*, pages 8546–8555. PMLR. 24, 35, 192

Petersen, F., Kuehne, H., Borgelt, C., and Deussen, O. (2022). Differentiable top-k classification learning. In *International Conference on Machine Learning*, pages 17656–17668. PMLR. 24, 35, 191, 202

Peyré, G., Cuturi, M., et al. (2019). Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607. 31, 32, 144, 147, 156, 157, 158

Pontryagin, L. S. (1987). *Mathematical theory of optimal processes*. CRC press. 18, 45, 116

Press, O., Smith, N. A., and Lewis, M. (2021). Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*. 168

Prillo, S. and Eisenschlos, J. (2020). Softsort: A continuous relaxation for the argsort operator. In *International Conference on Machine Learning*, pages 7793–7802. PMLR. 24, 192

Qian, Y., Lee, J., Duddu, S. M. K., Dai, Z., Brahma, S., Naim, I., Lei, T., and Zhao, V. Y. (2022). Multi-vector retrieval as sparse alignment. *arXiv preprint arXiv:2211.01267*. 24, 35, 191

Queiruga, A., Erichson, N. B., Hodgkinson, L., and Mahoney, M. W. (2021). Stateful ODE-nets using basis function expansions. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21770–21781. Curran Associates, Inc. 26, 69

Queiruga, A. F., Erichson, N. B., Taylor, D., and Mahoney, M. W. (2020). Continuous-in-depth neural networks. *arXiv preprint arXiv:2008.02389*. 116

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training. 19, 166

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9. 31, 144, 176

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551. 168

Renardy, M. and Rogers, R. C. (2006). *An introduction to partial differential equations*, volume 13. Springer Science & Business Media. 149

Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Susano Pinto, A., Keysers, D., and Houlsby, N. (2021). Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595. 23, 192, 202, 210

Rolínek, M., Musil, V., Paulus, A., Vlastelica, M., Michaelis, C., and Martius, G. (2020). Optimizing rank-based metrics with blackbox differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7620–7630. 191

Rosenblatt, F. et al. (1962). *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*, volume 55. Spartan books Washington, DC. 15

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*. 118, 160

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536. 116

Runckel, H.-J. (1969). Zeros of entire functions. *Transactions of the American Mathematical Society*, 143:343–362. 123, 133

Rusch, T. K. and Mishra, S. (2021). Coupled oscillatory recurrent neural network (co{rnn}): An accurate and (gradient) stable architecture for learning long time dependencies. In *International Conference on Learning Representations*. 117

Ruthotto, L. and Haber, E. (2019). Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, pages 1–13. 44, 116, 146

Sander, M. E., Ablin, P., Blondel, M., and Peyré, G. (2021). Momentum residual neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9276–9287. PMLR. 5, 45, 146

Sander, M. E., Ablin, P., Blondel, M., and Peyré, G. (2022a). Sinkformers: Transformers with doubly stochastic attention. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 3515–3530. PMLR. 5, 21, 22, 29, 44, 76, 179

Sander, M. E., Ablin, P., and Peyré, G. (2022b). Do residual neural networks discretize neural ordinary differential equations? In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 36520–36532. Curran Associates, Inc. 5, 26, 68, 69, 70

Sander, M. E., Giryes, R., Suzuki, T., Blondel, M., and Peyré, G. (2024). How do transformers perform in-context autoregressive learning ? In *Forty-first International Conference on Machine Learning*. 5, 23

Sander, M. E., Puigcerver, J., Djolonga, J., Peyré, G., and Blondel, M. (2023). Fast, differentiable and sparse top-k: a convex analysis perspective. In *International Conference on Machine Learning*, pages 29919–29936. PMLR. 5

Santambrogio, F. (2017). {Euclidean, metric, and Wasserstein} gradient flows: an overview. *Bulletin of Mathematical Sciences*, 7(1):87–154. 32, 149, 150, 157

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*. 23, 35, 190, 192

Singer, A. (2006). From graph to manifold laplacian: The convergence rate. *Applied and Computational Harmonic Analysis*, 21(1):128–134. 146, 151, 159

Sinkhorn, R. (1964). A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879. 21, 31, 144, 147

Sinkhorn, R. (1967). Diagonal equivalence to matrices with prescribed row and column sums. *The American Mathematical Monthly*, 74(4):402–405. 192

Song, M., Montanari, A., and Nguyen, P. (2018). A mean field view of the landscape of two-layers neural networks. *Proceedings of the National Academy of Sciences*, 115:E7665–E7671. 146

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32. 13

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*. 13

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Highway networks. *arXiv preprint arXiv:1505.00387*. 14

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. (2024). Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063. 168

Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852. 36, 203, 210

Sun, Q., Tao, Y., and Du, Q. (2018). Stochastic training of residual networks: a differential equation viewpoint. *arXiv preprint arXiv:1812.00174*. 44, 116, 146

Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., and Liang, J. (2016). Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312. 125, 127

Tay, Y., Bahri, D., Yang, L., Metzler, D., and Juan, D.-C. (2020). Sparse sinkhorn attention. In *International Conference on Machine Learning*, pages 9438–9447. PMLR. 146

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. (2023). Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*. 20, 21

Teh, Y., Doucet, A., and Dupont, E. (2019). Augmented neural odes. *Advances in Neural Information Processing Systems 32 (NIPS 2019)*, 32(2019). 18, 31, 42, 44, 71, 116, 117, 121, 124, 146

Teshima, T., Tojo, K., Ikeda, M., Ishikawa, I., and Oono, K. (2020). Universal approximation property of neural ordinary differential equations. *arXiv preprint arXiv:2012.02414*. 17, 26, 31, 42, 44, 69, 117

Thorpe, M. and van Gennip, Y. (2022). Deep limits of residual neural networks. *Research in the Mathematical Sciences*, 10:6. 70

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288. 127

Tibshirani, R. J. (2017). Dykstra's algorithm, admm, and coordinate descent: Connections, insights, and extensions. *Advances in Neural Information Processing Systems*, 30. 200, 208

Tierney, L., Kass, R. E., and Kadane, J. B. (1989). Fully exponential laplace approximations to expectations and variances of nonpositive functions. *Journal of the american statistical association*, 84(407):710–716. 151

Ting, D., Huang, L., and Jordan, M. (2011). An analysis of the convergence of graph laplacians. *arXiv preprint arXiv:1101.5435*. 146

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*. 20, 166, 168

Touvron, H., Vedaldi, A., Douze, M., and Jegou, H. (2019). 114

Tropp, J. A. (2012). User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12:389–434. 107

Tsai, Y.-H. H., Bai, S., Yamada, M., Morency, L.-P., and Salakhutdinov, R. (2019). Transformer dissection: a unified understanding of transformer's attention via the lens of kernel. *arXiv preprint arXiv:1908.11775*. 212

Tu, Z., Talebi, H., Zhang, H., Yang, F., Milanfar, P., Bovik, A., and Li, Y. (2022). Maxvit: Multi-axis vision transformer. In *European conference on computer vision*, pages 459–479. Springer. 20

Vardi, G. (2023). On the implicit bias in deep-learning algorithms. *Communications of the ACM*, 66:86–93. 26, 69

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008. 15, 19, 25, 31, 42, 68, 144, 145, 166, 168, 177, 178, 179, 187, 188

Verma, A. (2000). An introduction to automatic differentiation. *Current Science*, pages 804–807. 116

Vershynin, R. (2018). *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge. 98

Von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. (2023a). Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR. 23, 169

Von Oswald, J., Niklasson, E., Schlegel, M., Kobayashi, S., Zucchet, N., Scherrer, N., Miller, N., Sandler, M., y Arcas, B. A., Vladymyrov, M., Pascanu, R., and Sacramento, J. (2023b). Uncovering mesa-optimization algorithms in transformers. 23, 33, 34, 166, 169, 171, 180

Vuckovic, J., Baratin, A., and Combes, R. T. d. (2021). On the regularity of attention. *arXiv preprint arXiv:2102.05628*. 21, 146

Wang, H., Ma, S., Dong, L., Huang, S., Zhang, D., and Wei, F. (2022). DeepNet: Scaling transformers to 1,000 layers. *arXiv:2203.00555*. 68

Wang, L., Ye, J., Zhao, Y., Wu, W., Li, A., Song, S. L., Xu, Z., and Kraska, T. (2018). Superneurons: Dynamic gpu memory management for training deep neural networks. In *Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 41–53. 27, 30, 42, 44, 114, 150

Wang, M. and E, W. (2024). Understanding the expressive power and mechanisms of transformer for sequence modeling. 22

Warmuth, M. K. and Kuzmin, D. (2008). Randomized online pca algorithms with regret bounds that are logarithmic in the dimension. *Journal of Machine Learning Research*, 9(Oct):2287–2320. 194

Weinan, E. (2017a). A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11. 17

Weinan, E. (2017b). A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11. 44, 116, 146

Weinan, E., Han, J., and Li, Q. (2019). A mean-field optimal control formulation of deep learning. *Research in the Mathematical Sciences*, 6(1):10. 44, 116, 146

Wightman, R., Touvron, H., and Jégou, H. (2021). Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*. 25, 42

Wiseman, S. and Rush, A. M. (2016). Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960*. 190

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2019). Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*. 31, 144

Woodworth, B., Gunasekar, S., Lee, J. D., Moroshko, E., Savarese, P., Golan, I., Soudry, D., and Srebro, N. (2020). Kernel and rich regimes in overparametrized models. In *Conference on Learning Theory*, pages 3635–3673. PMLR. 171

Wormell, C. L. and Reich, S. (2021). Spectral convergence of diffusion maps: Improved error bounds and an alternative normalization. *SIAM Journal on Numerical Analysis*, 59(3):1687–1734. 146

Wortsman, M., Lee, J., Gilmer, J., and Kornblith, S. (2023). Replacing softmax with relu in vision transformers. *arXiv preprint arXiv:2309.08586*. 21

Wu, L., Wang, Q., and Ma, C. (2019). Global convergence of gradient descent for deep linear residual networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32, pages 13389–13398. Curran Associates, Inc. 70

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920. 33, 152

Xie, Y., Dai, H., Chen, M., Dai, B., Zhao, T., Zha, H., Wei, W., and Pfister, T. (2020). Differentiable top-k with optimal transport. *Advances in Neural Information Processing Systems*, 33:20520–20531. 192

Yang, G. and Schoenholz, S. (2017). Mean field residual networks: On the edge of chaos. *Advances in neural information processing systems*, 30. 47

Yao, Y., Rosasco, L., and Caponnetto, A. (2007). On early stopping in gradient descent learning. *Constructive Approximation*, 26:289–315. 175

Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S. J., and Kumar, S. (2019). Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*. 22, 144

Zeng, X. and Figueiredo, M. A. (2014). The ordered weighted l_1 norm: Atomic formulation, projections, and algorithms. *arXiv preprint arXiv:1409.4271*. 197, 198

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. (2021). Scaling vision transformers. *arXiv preprint arXiv:2106.04560*. 31, 144, 145, 155

Zhang, A., Chan, A., Tay, Y., Fu, J., Wang, S., Zhang, S., Shao, H., Yao, S., and Lee, R. K.-W. (2021). On orthogonality constraints for transformers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, volume 2, pages 375–382. Association for Computational Linguistics. 173

Zhang, H., Dauphin, Y. N., and Ma, T. (2019a). Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations*. 71

Zhang, H., Gao, X., Unterman, J., and Arodz, T. (2020a). Approximation capabilities of neural ODEs and invertible residual networks. In Daumé III, H. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 11086–11095. PMLR. 17, 26, 69

Zhang, H., Gao, X., Unterman, J., and Arodz, T. (2020b). Approximation capabilities of neural odes and invertible residual networks. In *International Conference on Machine Learning*, pages 11086–11095. PMLR. 31, 117

Zhang, R., Frei, S., and Bartlett, P. L. (2023). Trained transformers learn linear models in-context. *arXiv preprint arXiv:2306.09927*. 23, 168, 169, 170, 172

Zhang, T., Yao, Z., Gholami, A., Keutzer, K., Gonzalez, J., Biros, G., and Mahoney, M. (2019b). Anodev2: A coupled neural ode evolution framework. *arXiv preprint arXiv:1906.04596*. 116

Zhao, H., Jiang, L., Jia, J., Torr, P., and Koltun, V. (2020). Point transformer. *arXiv preprint arXiv:2012.09164*. 31, 144, 145

Zhou, Y., Lei, T., Liu, H., Du, N., Huang, Y., Zhao, V., Dai, A., Chen, Z., Le, Q., and Laudon, J. (2022). Mixture-of-experts with expert choice routing. *arXiv preprint arXiv:2202.09368*. 192, 203, 210

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232. 27, 42, 44, 114, 150

Zhuang, J., Dvornek, N., Li, X., Tatikonda, S., Papademetris, X., and Duncan, J. (2020). Adaptive checkpoint adjoint method for gradient estimation in neural ode. In *International Conference on Machine Learning*, pages 11639–11649. PMLR. 46

Zhuang, J., Dvornek, N. C., Tatikonda, S., and Duncan, J. S. (2021). Mali: A memory efficient and reverse accurate integrator for neural odes. *arXiv preprint arXiv:2102.04668*. 18, 44

Ziegler, G. M. (2012). *Lectures on polytopes*, volume 152. Springer Science & Business Media. 194

Zou, D., Long, P. M., and Gu, Q. (2020a). On the global convergence of training deep linear resnets. *arXiv preprint arXiv:2003.01094*. 27, 47, 54

Zou, D., Long, P. M., and Gu, Q. (2020b). On the global convergence of training deep linear ResNets. In *International Conference on Learning Representations*. 70

Zweig, A. and Bruna, J. (2021). A functional perspective on learning symmetric functions with neural networks. In *International Conference on Machine Learning*, pages 13023–13032. PMLR. 21, 146