Momentum Residual Neural Networks

Michael E. Sander, Pierre Ablin, Mathieu Blondel, Gabriel Peyré

Momentum Residual Neural Networks

Michael E. Sander [1 2] **Pierre Ablin** [1 2] **Mathieu Blondel** [3] **Gabriel Peyré** [1 2]

Abstract

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 paper, 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.

1. Introduction

Problem setup. As a particular instance of deep learning (LeCun et al., 2015; Goodfellow et al., 2016), residual neural networks (He et al., 2016, 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 paper 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)

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

we define its momentum counterpart, which iterates

$$\begin{cases} v_{n+1} = \gamma v_n + (1-\gamma) f(x_n, \theta_n) \\ x_{n+1} = x_n + v_{n+1}, \end{cases} \tag{2}$$

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.

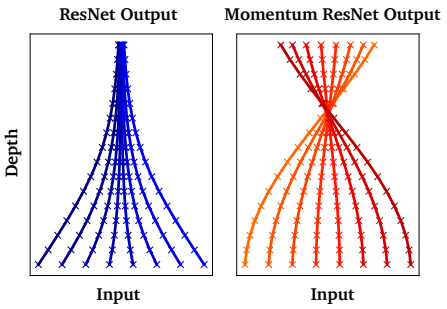

Figure 1. **Comparison of the dynamics** of a ResNet (left) and a Momentum ResNet with $\gamma = 0.9$ (right) with **tied weights** between layers, $\theta_n = \theta$ for all n. The evolution of the activations at each layer is shown (depth 15). Models try to learn the mapping $x \mapsto -x^3$ in $\mathbb{R}$. 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 architec-

[1]Ecole Normale Supérieure, DMA, Paris, France [2]CNRS, France [3]Google Research, Brain team. Correspondence to: Michael Sander <michael.sander@ens.fr>, Pierre Ablin <pierre.ablin@ens.fr>, Mathieu Blondel <mblondel@google.com>, Gabriel Peyré <gabriel.peyre@ens.fr>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

ture which uses residual blocks as building blocks (cf. in Section 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 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 3.3). We also investigate the approximation capabilities of Momentum ResNets, seen in the continuous limit as second-order ODEs (Section 4). We first show in Proposition 3 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 1 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 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 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 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`.

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 & Walther, 2008). 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 & 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 & Ruthotto, 2017; 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, 2017; Chen et al., 2018; Teh et al., 2019; Sun et al., 2018; Weinan et al., 2019; Lu et al., 2018; Ruthotto & 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, 2018), 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., 2019; 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. (2020) 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 & 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 & 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 $\mathbb{C}$.

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.

3.1. Momentum ResNets

Adding a momentum term in the ResNet equations. For **any** ResNet which iterates (1), we define its Momentum counterpart, which iterates (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 paper, 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 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 (2) is inverted through

$$\begin{cases} x_n = x_{n+1} - v_{n+1}, \\ v_n = \frac{1}{\gamma}\left(v_{n+1} - (1-\gamma) f(x_n, \theta_n)\right), \end{cases} \tag{3}$$

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 (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. We further describe the procedure in Appendix C. 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 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., 2016). 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 (1) into (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 1 summarizes the properties of reversible residual architectures. We discuss in further details the differences between RevNets and Momentum ResNets in sections 3.3 and 5.3.

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 $f(x,\theta) = W_2^T\sigma(W_1x+b)$, 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 2.

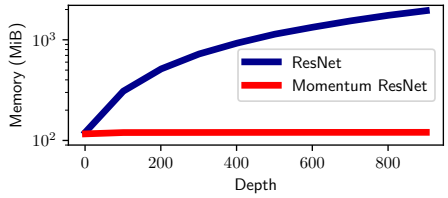

Figure 2. **Comparison of memory needed** (calculated using a profiler) for computing gradients of the loss, with ResNets (activations are stored) and Momentum ResNets (activations are not stored). We set $n_{batch} = 500$, $d = 500$ and $\gamma = 1 - \frac{1}{50k}$ at each depth. Momentum ResNets give a nearly constant memory footprint.

As another example, consider a ResNet-152 (He et al., 2016) which can be used for ImageNet classification (Deng et al., 2009). 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.

3.3. The role of momentum

When γ is set to 0 in (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 (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

$$v_{n+1} = v_n + \varphi(x_n, \theta_n), \quad x_{n+1} = x_n + \psi(v_{n+1}, \theta_n'), \quad (4)$$

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) 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 1 (Instability of fixed points)**.** *Let (v^*, x^*) a fixed point of the RevNet iteration (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*

$J(A,B) = \begin{pmatrix} \mathrm{Id}_d & A \\ B & \mathrm{Id}_d + BA \end{pmatrix}$. *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), 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 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 1 has a continuous counterpart. Indeed, in the continuous limit, (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).

3.4. Momentum ResNets as continuous models

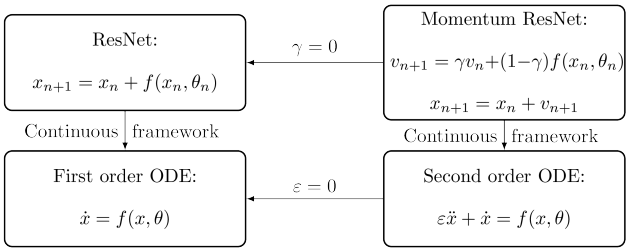

Figure 3. **Overview of the four different paradigms.**

Neural ODEs: ResNets as first-order ODEs. The ResNets equation (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 (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 3 sums up these ideas.

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.

4.1. Representation capabilities of first-order ODEs

We consider the first-order model

$$\dot{x} = f(x,\theta) \quad \text{with} \quad x(0) = x_0. \tag{5}$$

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 (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 (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 (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 B.5).

4.2. Representation capabilities of second-order ODEs

We consider the second-order model for which we recall that Momentum ResNets are a discretization:

$$\varepsilon\ddot{x} + \dot{x} = f(x,\theta) \quad \text{with} \quad (x(0),\dot{x}(0)) = (x_0, v_0). \tag{6}$$

In Section 3.3, we showed that Momentum ResNets generalize existing models when setting $\gamma = 0$ or 1. We now state the continuous counterparts of these results. Recall that $\frac{1}{1-\gamma} = \varepsilon$. When $\varepsilon \to 0$, we recover the first-order model.

Proposition 2 (Continuity of the solutions)**.** *We let x^* (resp. x_ε) be the solution of (5) (resp. (6)) on $[0,T]$, with initial conditions $x^*(0) = x_\varepsilon(0) = x_0$ and $\dot{x}_\varepsilon(0) = v_0$. Then $\|x_\varepsilon - x^*\|_\infty \to 0$ as $\varepsilon \to 0$.*

The proof of this result relies on the implicit function theorem and can be found in Appendix A.1. Note that Proposition 2 is true whatever the initial speed v_0. When $\varepsilon \to +\infty$, one needs to rescale f to study the asymptotics: the solution of $\ddot{x} + \frac{1}{\varepsilon}\dot{x} = f(x,\theta)$ converges to the solution of

$\ddot{x} = f(x, \theta)$ (see details in Appendix B.1). These results show that in the continuous regime, Momentum ResNets also interpolate between $\dot{x} = f(x, \theta)$ and $\ddot{x} = f(x, \theta)$.

Representation capabilities of a model (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 (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 (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 (6) can represent any mapping, as proved in Appendix 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 3 (Momentum ResNets are at least as general)**.** *There exists a function $\hat{f}$ such that for all x solution of (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 (6) (see details in Appendix B.4). This supports the claim that the set of representable mappings increases with ε.

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

$$\varepsilon\ddot{x} + \dot{x} = \theta x \quad \text{with} \quad (x(0), \dot{x}(0)) = (x_0, 0), \qquad (7)$$

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 (Solution of (7))**.** *At time 1, (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 (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 $\varepsilon \to 0$, Proposition 2 shows that $\Psi_\varepsilon(\theta) \to \Psi_0(\theta) = \exp\theta$. The range of the matrix exponential is indeed the set of representable mappings of a first order linear model

$$\dot{x} = \theta x \quad \text{with} \quad x(0) = x_0 \qquad (8)$$

and this range is known (Andrica & 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 (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 A.4 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 (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.

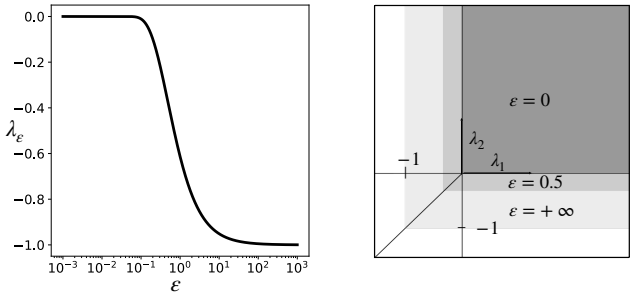

Figure 4. Left: **Evolution of λ_ε defined in Theorem 1**. λ_ε is non increasing, stays close to 0 when $\varepsilon \ll 1$ and close to -1 when $\varepsilon \geq 2$. Right: **Evolution of the real eigenvalues** λ_1 and λ_2 of representable matrices in $\mathrm{D}_d^{\mathbb{C}}(\mathbb{R})$ by (7) when $d = 2$ 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 A.4, 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 con-

veniently 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 1 (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 (7) **if and only if** $\forall\lambda \in \mathrm{Sp}(D)$ such that $\lambda < \lambda_\varepsilon$, λ is of even multiplicity order.*

Theorem 1 is illustrated in Figure 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 (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 1 shows that $\alpha_\varepsilon D$ is represented by a second-order model (7).

5. Experiments

We now demonstrate the applicability of Momentum ResNets through experiments. We used Pytorch and Nvidia Tesla V100 GPUs.

5.1. Point clouds separation

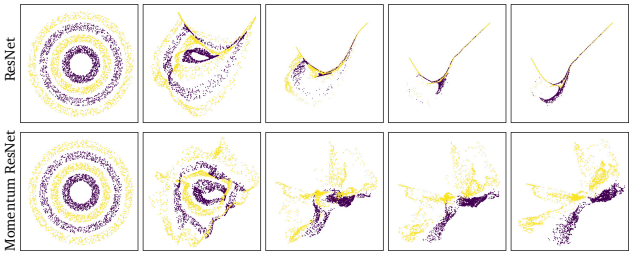

Figure 5. **Separation of four nested rings** using a ResNet (upper row) and a Momentum ResNet (lower row). From left to right, each figure represents the point clouds transformed at layer $3k$. The ResNet fails whereas the Momentum ResNet succeeds.

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 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.

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., 2009). 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 D.

Table 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., 2016) 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 2.

Effect of the momentum term γ. Theorem 1 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 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 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 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 7 and illustrate how Momentum ResNets can benefit from increased batch size, especially for very deep models. We also

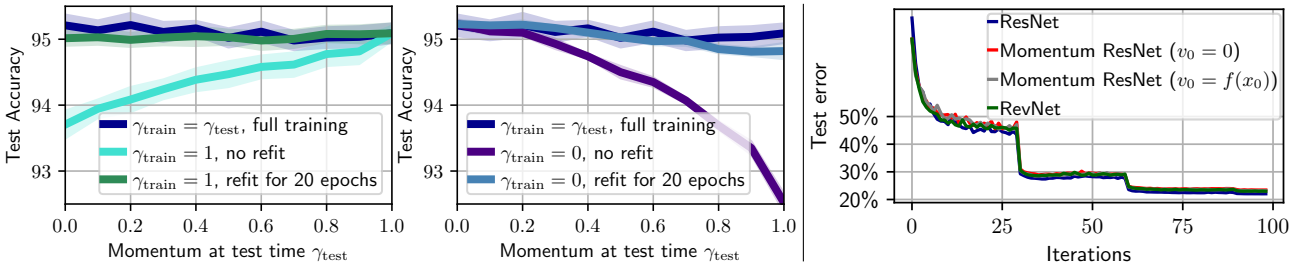

Figure 6. Upper row: **Robustness of final accuracy w.r.t** γ when training Momentum ResNets 101 on CIFAR-10. We train the networks with a momentum γ_{train} and evaluate their accuracy with a different momentum γ_{test} at test time. We optionally refit the networks for 20 epochs. We recall that $\gamma_{\text{train}} = 0$ 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.

show in Figure 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 3.2)).

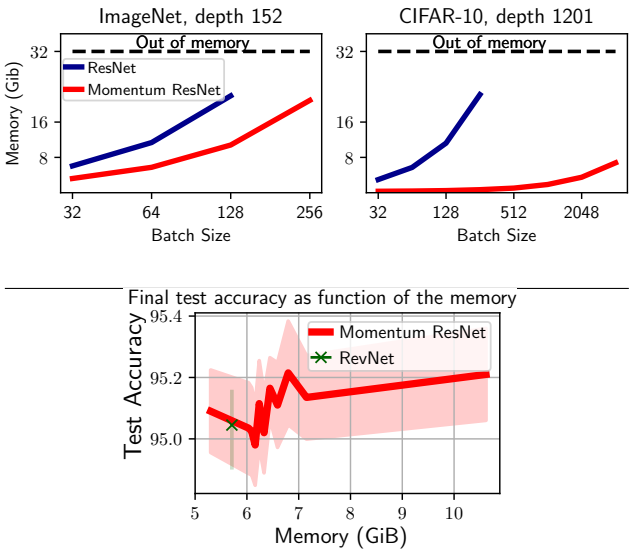

Figure 7. Upper row: **Memory used** (using a profiler) for a ResNet and a Momentum ResNet on one training epoch, as a function of the batch size. Lower row: **Final test accuracy** as a function of the memory used (per epoch) for training Momentum ResNets-101 on CIFAR-10.

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 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.

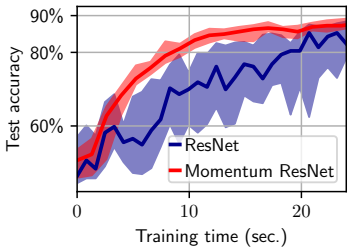

Figure 8. **Accuracy** as a function of time on `hymenoptera` when fine-tuning a ResNet-152 and a Momentum ResNet-152 with batch sizes of 2 and 4, respectively, as permitted by memory.

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 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 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), 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 (5) and the other one to the second-order ODE (6). Training was based on the odeint function imple-

[1] https://www.kaggle.com/ajayrana/hymenoptera-data

mented by Chen et al. (2018). Figure 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.

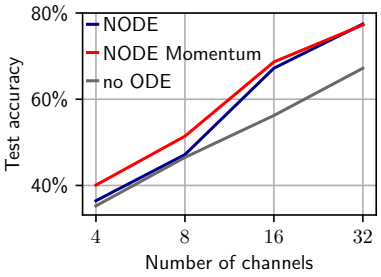

Figure 9. **Accuracy after 120 iterations on CIFAR-10** with or without momentum, when varying the number of channels.

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 & 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 & 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 D. The next Figure 10 shows the test loss of the different methods, when the depth of the networks varies.

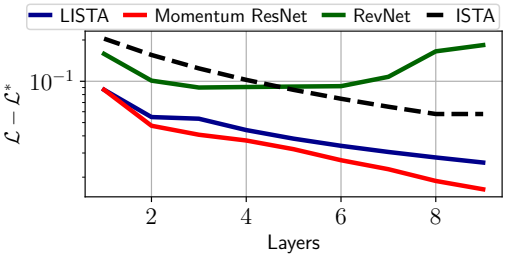

Figure 10. **Evolution of the test loss** for different models as a function of depth in the Learned-ISTA (LISTA) framework.

As predicted by Proposition 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 paper 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 paper, the latter is not possible with existing invertible residual networks, although crucial in the learning to optimize setting.

Acknowledgments

This work was granted access to the HPC resources of IDRIS under the allocation 2020-[AD011012073] made by GENCI. This work was supported in part by the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR19-P3IA-0001 (PRAIRIE 3IA Institute). This work was supported in part by the European Research Council (ERC project NORIA). The authors would like to thank David Duvenaud and Dougal Maclaurin for their helpful feedbacks. M. S. thanks Pierre Rizkallah and Pierre Roussillon for fruitful discussions.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Andrica, D. and Rohan, R.-A. The image of the exponential map and some applications. In *Proc. 8th Joint Conference on Mathematics and Computer Science MaCS, Komarno, Slovakia*, pp. 3–14, 2010.

Bauer, F. L. Computational graphs and rounding error. *SIAM Journal on Numerical Analysis*, 11(1):87–96, 1974.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18, 2018.

Behrmann, J., Grathwohl, W., Chen, R. T., Duvenaud, D., and Jacobsen, J.-H. Invertible residual networks. In *International Conference on Machine Learning*, pp. 573–582. PMLR, 2019.

Chang, B., Meng, L., Haber, E., Ruthotto, L., Begert, D., and Holtham, E. Reversible architectures for arbitrarily deep residual neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in neural information processing systems*, pp. 6571–6583, 2018.

Chun, I. Y., Huang, Z., Lim, H., and Fessler, J. Momentum-net: Fast and convergent iterative neural network for inverse problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Conway, J. B. *Functions of one complex variable II*, volume 159. Springer Science & Business Media, 2012.

Craven, T. and Csordas, G. Iterated laguerre and turán inequalities. *J. Inequal. Pure Appl. Math*, 3(3):14, 2002.

Culver, W. J. On the existence and uniqueness of the real logarithm of a matrix. *Proceedings of the American Mathematical Society*, 17(5):1146–1151, 1966.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Daubechies, I., Defrise, M., and De Mol, C. 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, 2004.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dryanov, D. and Rahman, Q. Approximation by entire functions belonging to the laguerre–polya class. *Methods and Applications of Analysis*, 6(1):21–38, 1999.

Gantmacher, F. R. The theory of matrices. *Chelsea, New York*, Vol. I, 1959.

Gholami, A., Keutzer, K., and Biros, G. Anode: Unconditionally accurate memory-efficient gradients for neural odes. *arXiv preprint arXiv:1902.10298*, 2019.

Gomez, A. N., Ren, M., Urtasun, R., and Grosse, R. B. 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. (eds.), *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Gregor, K. and LeCun, Y. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conferenceon machine learning*, pp. 399–406, 2010.

Griewank, A. and Walther, A. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

Gusak, J., Markeeva, L., Daulbaev, T., Katrutsa, A., Cichocki, A., and Oseledets, I. Towards understanding normalization in neural odes. *arXiv preprint arXiv:2004.09222*, 2020.

Haber, E. and Ruthotto, L. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2017.

Hairer, E., Lubich, C., and Wanner, G. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.

Hartfiel, D. J. Dense sets of diagonalizable matrices. *Proceedings of the American Mathematical Society*, 123(6): 1669–1672, 1995.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference*

on Computer Vision and Pattern Recognition, pp. 9729–9738, 2020.

Jacobsen, J.-H., Smeulders, A. W., and Oyallon, E. i-revnet: Deep invertible networks. In *International Conference on Learning Representations*, 2018.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*, 6(2):8, 2019.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*, 5, 2010.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Levin, B. Y. *Lectures on entire functions*, volume 150. American Mathematical Soc., 1996.

Li, H., Yang, Y., Chen, D., and Lin, Z. Optimization algorithm inspired deep neural network structure design. In *Asian Conference on Machine Learning*, pp. 614–629. PMLR, 2018.

Li, Q., Lin, T., and Shen, Z. Deep learning via dynamical systems: An approximation perspective. *arXiv preprint arXiv:1912.10382*, 2019.

Lu, Y., Zhong, A., Li, Q., and Dong, B. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *International Conference on Machine Learning*, pp. 3276–3285. PMLR, 2018.

Maclaurin, D., Duvenaud, D., and Adams, R. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pp. 2113–2122. PMLR, 2015.

Martens, J. and Sutskever, I. Training deep and recurrent networks with hessian-free optimization. In *Neural networks: Tricks of the trade*, pp. 479–535. Springer, 2012.

Massaroli, S., Poli, M., Park, J., Yamashita, A., and Asama, H. Dissecting neural odes. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 3952–3963. Curran Associates, Inc., 2020.

Nguyen, T. M., Baraniuk, R. G., Bertozzi, A. L., Osher, S. J., and Wang, B. Momentumrnn: Integrating momentum into recurrent neural networks. *arXiv preprint arXiv:2006.06919*, 2020.

Norcliffe, A., Bodnar, C., Day, B., Simidjievski, N., and Liò, P. On second order behaviour in augmented neural odes. *arXiv preprint arXiv:2006.07220*, 2020.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Peng, C., Zhang, X., Yu, G., Luo, G., and Sun, J. Large kernel matters–improve semantic segmentation by global convolutional network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4353–4361, 2017.

Perko, L. *Differential equations and dynamical systems*, volume 7. Springer Science & Business Media, 2013.

Pontryagin, L. S. *Mathematical theory of optimal processes*. Routledge, 2018.

Queiruga, A. F., Erichson, N. B., Taylor, D., and Mahoney, M. W. Continuous-in-depth neural networks. *arXiv preprint arXiv:2008.02389*, 2020.

Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Runckel, H.-J. Zeros of entire functions. *Transactions of the American Mathematical Society*, 143:343–362, 1969.

Rusch, T. K. and Mishra, S. Coupled oscillatory recurrent neural network (co{rnn}): An accurate and (gradient) stable architecture for learning long time dependencies. In *International Conference on Learning Representations*, 2021.

Ruthotto, L. and Haber, E. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, pp. 1–13, 2019.

Sun, Q., Tao, Y., and Du, Q. Stochastic training of residual networks: a differential equation viewpoint. *arXiv preprint arXiv:1812.00174*, 2018.

Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., and Liang, J. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312, 2016.

Teh, Y., Doucet, A., and Dupont, E. Augmented neural odes. *Advances in Neural Information Processing Systems 32 (NIPS 2019)*, 32(2019), 2019.

Teshima, T., Tojo, K., Ikeda, M., Ishikawa, I., and Oono, K. Universal approximation property of neural ordinary differential equations. *arXiv preprint arXiv:2012.02414*, 2020.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Touvron, H., Vedaldi, A., Douze, M., and Jegou, H. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

Verma, A. An introduction to automatic differentiation. *Current Science*, pp. 804–807, 2000.

Wang, L., Ye, J., Zhao, Y., Wu, W., Li, A., Song, S. L., Xu, Z., and Kraska, T. 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*, pp. 41–53, 2018.

Weinan, E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

Weinan, E., Han, J., and Li, Q. A mean-field optimal control formulation of deep learning. *Research in the Mathematical Sciences*, 6(1):10, 2019.

Zhang, H., Gao, X., Unterman, J., and Arodz, T. Approximation capabilities of neural odes and invertible residual networks. In *International Conference on Machine Learning*, pp. 11086–11095. PMLR, 2020.

Zhang, T., Yao, Z., Gholami, A., Keutzer, K., Gonzalez, J., Biros, G., and Mahoney, M. Anodev2: A coupled neural ode evolution framework. *arXiv preprint arXiv:1906.04596*, 2019.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

Appendix

In Section A we give the proofs of all the Propositions and the Theorem. In Section B we give other theoretical results to validate statements made in the paper. Section C presents the algorithm from Maclaurin et al. (2015). Section D gives details for the experiments in the paper. We derive the formula for backpropagation in Momentum ResNets in Section E. Finally, we present additional figures in Section F.

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 $z \in \mathbb{C}$.

A.0. Instability of fixed points – Proof of Proposition 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

$$\Psi(v, x) = (v^* + \delta + A\varepsilon, x^* + \varepsilon + B(\delta + A\varepsilon)) \tag{9}$$

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

$$u + Aw = \lambda u \tag{10}$$
$$w + Bu + BAw = \lambda w \tag{11}$$

We start by showing that $\lambda \neq 1$ by contradiction. Indeed, if $\lambda = 1$, then (10) gives $Aw = 0$, which implies $w = 0$ since A is invertible. Then, (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 (10) gives $Aw = (\lambda - 1)u$, and multiplying (11) by A on the left gives

$$\lambda ABu = (\lambda - 1)^2 u \tag{12}$$

We also cannot have $\lambda = 0$, since it would imply $u = 0$. Then, dividing (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, (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 (11): λ is indeed an eigenvalue of $J(A, B)$. □

A.1. Momentum ResNets in the limit $\varepsilon \to 0$ – Proof of Proposition 2

Proof. We take $T = 1$ without loss of generality. We are going to use the implicit function theorem. Note that x_ε is solution of (6) if and only if $(x_\varepsilon, v_\varepsilon = \dot{x_\varepsilon})$ is solution of

$$\begin{cases} \dot{x} &= v, \quad x(0) = x_0 \\ \varepsilon\dot{v} &= f(x, \theta) - v, \quad v(0) = v_0. \end{cases}$$

Consider for $u = (x, v) \in (x_0, v_0) + C_0^\infty([0, 1], \mathbb{R}^d)^2$

$$\Psi(u, \varepsilon) = \left(x_0 - x + \int_0^t v, \int_0^t (f(x, \theta) - v) - \varepsilon v + \varepsilon v_0 \right),$$

so that x_ε is solution of (6) if and only if $u_\varepsilon = (x_\varepsilon, v_\varepsilon = \dot{x_\varepsilon})$ satisfies $\Psi(u_\varepsilon, \varepsilon) = 0$. Let $u^* = (x^*, \dot{x^*})$. One has $\Psi(u^*, 0) = 0$. Ψ is differentiable everywhere, and at $(u^*, 0)$ we have

$$\partial_u \Psi(u^*, 0)(x, v) = \left((\int_0^t v) - x, \int_0^t (\partial_x f(x*, \theta).x - v) \right).$$

$\partial_u \Psi(u^*, 0)$ is continuous, and it is invertible with continuous inverse because it is linear and continuous, and because $\partial_u \Psi(u^*, 0)(x, v) = 0$ if and only if

$$\begin{cases} \forall t \in [0, 1], x(t) = \int_0^t v \\ \forall t \in [0, 1], v(t) = \partial_x f(x^*(t), \theta(t)).x(t) \end{cases}$$

which is equivalent to

$$\begin{cases} \dot{x} = \partial f(x^*, \theta).x \\ x(0) = 0 \\ v = \dot{x}, \end{cases}$$

which is equivalent, because this equation is linear to $(x, v) = (0, 0)$. Using the implicit function theorem, we know that there exists two neighbourhoods $U \subset \mathbb{R}$ and $V \subset (x_0, v_0) + C_0^\infty([0, 1], \mathbb{R}^d)^2$ of 0 and u^* and a continuous function $\zeta : U \to V$ such that

$$\forall (u, \varepsilon) \in U \times V, \Psi(u, \varepsilon) = 0 \Leftrightarrow u = \zeta(\varepsilon)$$

This in particular ensures that x_ε converges uniformly to x^* as ε goes to 0 □

A.2. Momentum ResNets are more general than neural ODEs – Proof of Proposition 3

Proof. If x satisfies (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 $(x(0), \dot{x}(0)) = (x_0, f(x_0,\theta_0))$. □

A.3. Solution of (7) – Proof of Proposition 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 (7) by derivations. At time 1 this effectively gives $x(1) = \Psi_\varepsilon(\theta) x_0$.

A.4. Representable mappings for a Momentum ResNet with linear residual functions – Proof of Theorem 1

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 $\mathbb{C}$.

A.4.1. Surjectivity on $\mathbb{C}$ of f_ε

Lemma 1 (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 1 in the diagonalizable case.

A.4.2. THEOREM 1 IN THE DIAGONALIZABLE CASE

Proof. **Necessity** Suppose that D can be represented by a second-order model (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 1, 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 1 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 (7). □

We now state and demonstrate the general version of Theorem 1.

First, we need to demonstrate properties of the complex derivatives of the entire function f_ε.

A.4.3. The entire function f_ε has a derivative with no-zeros on $\mathbb{C} - \mathbb{R}$.

Lemma 2 (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 $G'_\varepsilon(z) = -2z f'_\varepsilon(-z^2)$ 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 & 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 & 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.

□

A.4.4. Theorem 1 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 (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 (7).

Theorem 2 (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 (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 (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:

$$(\lambda - z_k)^{a_k},\ z_k \in \mathbb{R}$$
$$(\lambda - z_k)^{b_k} \text{ and } (\lambda - \bar{z_k})^{b_k},\ z_k \in \mathbb{C} - \mathbb{R}.$$

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 2. 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 1), $\lambda_k = f_\varepsilon(z_k)$ 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 2, $f'_\varepsilon(z_k) \neq 0$
- 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

$$(\lambda - f_\varepsilon(z_k))^{b_k} \text{ and } (\lambda - f_\varepsilon(\bar{z_k}))^{b_k},\ z_k \in \mathbb{C} - \mathbb{R}$$
$$(\lambda - f_\varepsilon(z_k))^{a_k},\ z_k \in \mathbb{R},\ f_\varepsilon(z_k) \neq \lambda_\varepsilon$$
$$(\lambda - \lambda_\varepsilon)^{c_k} \text{ and } (\lambda - \lambda_\varepsilon)^{c_k}.$$

Let $Y \in \mathbb{R}^{d\times d}$ be such that its Jordan blocks are of the form

$$(\lambda - z_k)^{b_k} \text{ and } (\lambda - \bar{z_k})^{b_k},\ z_k \in \mathbb{C} - \mathbb{R},\ f'_\varepsilon(z_k) \neq 0$$
$$(\lambda - z_k)^{a_k},\ z_k \in \mathbb{R},\ f_\varepsilon(z_k) \neq \lambda_\varepsilon,\ f'_\varepsilon(z_k) \neq 0$$
$$(\lambda - z_k)^{2c_k},\ z_k \in \mathbb{R},\ f_\varepsilon(z_k) = \lambda_\varepsilon.$$

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 $f_\varepsilon(X) = A$. □

B. Additional theoretical results

B.1. On the convergence of the solution of a second order model when $\varepsilon \to \infty$

Proposition 5 (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 $\varepsilon \to +\infty$.

□

B.2. Universality of Momentum ResNets

Proposition 6 (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 $\varphi_t(x_0) = x_0 - v_0(e^{-t} - 1)$. □

B.3. Non-universality of Momentum ResNets when $v_0 = 0$

Proposition 7 (When $v_0 = 0$ there are mappings that cannot be learned if the equation is autonomous.)**.** *When $d = 1$, consider the autonomous ODE*

$$\begin{aligned} \varepsilon\ddot{x} + \dot{x} &= f(x) \\ (x(0), \dot{x}(0)) &= (x_0, 0) \end{aligned} \tag{13}$$

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 (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 $x_0 \in \mathbb{R}*$. □

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 8. *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

$$\ddot{x} + \frac{1}{\varepsilon}\dot{x} = (-\pi^2 - \frac{1}{4\varepsilon^2})x \tag{14}$$

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:

$$x(1) = -x_0 e^{-\frac{1}{2\varepsilon}}$$

□

B.5. Orientation preservation of first-order ODEs

Proposition 9 (The homeomorphisms represented by (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 (5), then h is in the connected component of the identity function on K for the $\|.\|_\infty$ topology.*

We first prove the following:

Lemma 3. *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 3. 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. □

B.6. On the linear mappings represented by autonomous first order ODEs in dimension 1

Consider the autonomous ODE

$$\dot{x} = f(x), \tag{15}$$

Theorem 3 (Linearity)**.** *Suppose $d = 1$. If (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 (3), 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 (3) as

$$\frac{\dot{x}}{f(x)} = 1. \tag{16}$$

Consider F a primitive of $\frac{1}{f}$. Integrating (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}. □

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 (15).

Proposition 10 (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 (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 11 illustrates the paradox. □

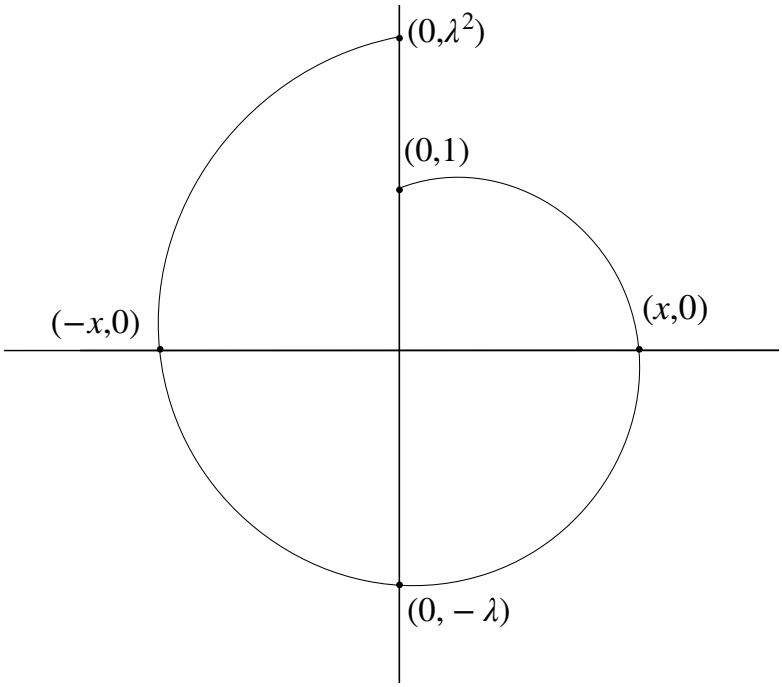

Figure 11. Illustration of Proposition 10. The points starting at $(0, 1)$ and $(0, \lambda^2)$ are distinct but their associated trajectories would have to intersect in $(x, 0)$, which is impossible.

C. Exact multiplication

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

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 (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. The entire procedure is illustrated in Algorithm 1 from Maclaurin et al. (2015).

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 $\lambda = 0.1$.

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 (2). Indeed, (2) writes explicitly

$$\begin{aligned} v_{n+1} &= \gamma v_n + (1-\gamma) f(x_n, \theta_n) \\ x_{n+1} &= x_n + (\gamma v_n + (1-\gamma) f(x_n, \theta_n)). \end{aligned} \tag{17}$$

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.

F. Additional figures

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.

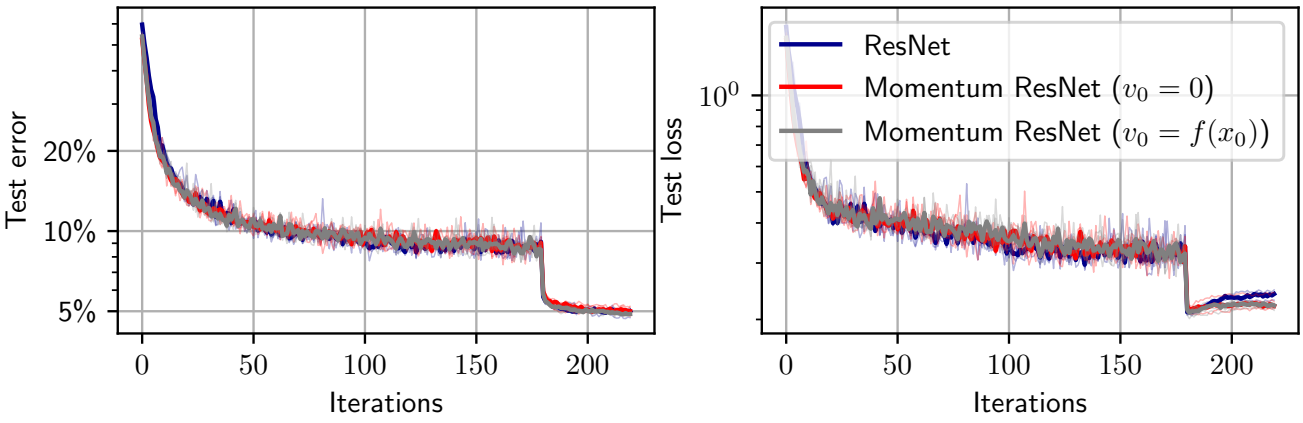

Figure 12. Test error and test loss as a function of depth on CIFAR-10 with a ResNet-101 and two Momentum ResNets-101.