A Dynamical System Perspective for Lipschitz Neural Networks

Laurent Meunier [* 1 2] **Blaise Delattre** [* 1 3] **Alexandre Araujo** [* 4] **Alexandre Allauzen** [1 5]

Abstract

The Lipschitz constant of neural networks has been established as a key quantity to enforce the robustness to adversarial examples. In this paper, we tackle the problem of building 1-Lipschitz Neural Networks. By studying Residual Networks from a continuous time dynamical system perspective, we provide a generic method to build 1-Lipschitz Neural Networks and show that some previous approaches are special cases of this framework. Then, we extend this reasoning and show that ResNet flows derived from convex potentials define 1-Lipschitz transformations, that lead us to define the *Convex Potential Layer* (CPL). A comprehensive set of experiments on several datasets demonstrates the scalability of our architecture and the benefits as an ℓ_2-provable defense against adversarial examples. Our code is available at `https://github.com/MILES-PSL/Convex-Potential-Layer`

1. Introduction

Modern neural networks have been known to be sensible against small, imperceptible and adversarially-chosen perturbations of their inputs (Biggio et al., 2013; Szegedy et al., 2014). This vulnerability has become a major issue as more and more neural networks have been deployed into production applications. Over the past decade, the research progress plays out like a cat-and-mouse game between the development of more and more powerful attacks (Goodfellow et al., 2015; Kurakin et al., 2016; Carlini et al., 2017; Croce et al., 2020) and the design of empirical defense mechanisms (Madry et al., 2018; Moosavi-Dezfooli et al., 2019; Cohen et al., 2019). Finishing the game calls for certified adversarial robustness (Raghunathan et al., 2018; Wong et al., 2018). While recent work devised defenses with theoretical guarantees against adversarial perturbations, they share the same limitation, *i.e.*, the tradeoffs between expressivity and robustness, and between scalability and accuracy.

A natural approach to provide robustness guarantees on a classifier is to enforce Lipschitzness properties. To achieve such properties, researchers mainly focused on two different kinds of approaches. The first one is based on randomization (Lecuyer et al., 2018; Cohen et al., 2019; Pinot et al., 2019) and consists in convolving the input with with a predefined probability distribution. While this approach offers some level of scalability (*i.e.*, currently the only certified defense on the ImageNet dataset), it suffers from significant impossibility results (Yang et al., 2020). A second approach consists in building 1-Lipschitz layers using specific linear transform (Cisse et al., 2017; Li et al., 2019b; Anil et al., 2019; Trockman et al., 2021; Singla & Feizi, 2021; Li et al., 2019b; Singla et al., 2021a). Knowing the Lipschitz constant of the network, it is then possible to compute a certification radius around any points.

A large line of work explored the interpretation of residual neural networks (He et al., 2016) as a parameter estimation problem of nonlinear dynamical systems (Haber et al., 2017; E, 2017; Lu et al., 2018). Reconsidering the ResNet architecture as an Euler discretization of a continuous dynamical system yields to the trend around Neural Ordinary Differential Equation (Chen et al., 2018). For instance, in the seminal work of Haber et al. (2017), the continuous formulation offers more flexibility to investigate the stability of neural networks during inference, knowing that the discretization will be then implemented by the architecture design. The notion of stability, in our context, quantifies how a small perturbation on the initial value impacts the trajectories of the dynamical system.

From this continuous and dynamical interpretation, we analyze the Lipschitzness property of Neural Networks. We then study the discretization schemes that can preserve the Lipschitzness properties. With this point of view, we can readily recover several previous methods that build 1-Lipschitz neural networks (Trockman et al., 2021; Singla & Feizi, 2021). Therefore, the dynamical system perspective offers a general and flexible framework to build Lip-

[*]Equal contribution [1]Miles Team, LAMSADE, Université Paris-Dauphine, PSL University, Paris, France [2]Meta AI Research, Paris, France [3]Foxstream, Lyon, France [4]INRIA, Ecole Normale Supérieure, CNRS, PSL University, Paris, France [5]ESPCI, Paris, France. Correspondence to: Laurent Meunier <laurent.meunier1995@gmail.com>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

schitz Neural Networks facilitating the discovery of new approaches. In this vein, we introduce convex potentials in the design of the Residual Network flow and show that this choice of parametrization yields to by-design 1-Lipschitz neural networks. At the very core of our approach lies a new 1-Lipschitz non-linear operator that we call *Convex Potential Layer* which allows us to adapt convex potential flows to the discretized case. These blocks enjoy the desirable property of stabilizing the training of the neural network by controlling the gradient norm, hence overcoming the exploding gradient issue. We experimentally demonstrate our approach by training large-scale neural networks on several datasets, reaching state-of-the art results in terms of under-attack and certified accuracy.

2. Background and Related Work

In this paper, we aim at devising *certified* defense mechanisms against adversarial attacks, in the following, we formally define an adversarial attacks and a robustness certificate. We consider a classification task from an input space $\mathcal{X} \subset \mathbb{R}^d$ to a label space $\mathcal{Y} := \{1, \dots, K\}$. To this end, we aim at learning a classifier function $\mathbf{f} := (f_1, \dots, f_K) : \mathcal{X} \to \mathbb{R}^K$ such that the predicted label for an input x is $\operatorname{argmax}_k f_k(x)$. For a given couple input-label (x, y), we say that x is correctly classified if $\operatorname{argmax}_k f_k(x) = y$.

Definition 1 (Adversarial Attacks). *Let $x \in \mathcal{X}$, $y \in \mathcal{Y}$ the label of x and let $\mathbf{f}$ be a classifier. An adversarial attack at level ε is a perturbation τ s.t. $\|\tau\| \le \varepsilon$ such that:*

$$\operatorname*{argmax}_k f_k(x + \tau) \neq y$$

Let us now define the notion of robust certification. For $x \in \mathcal{X}$, $y \in \mathcal{Y}$ the label of x and let $\mathbf{f}$ be a classifier, a classifier $\mathbf{f}$ is said to be *certifiably robust at radius* $\varepsilon \ge 0$ at point x if for all τ such that $\|\tau\| \le \varepsilon$:

$$\operatorname*{argmax}_k f_k(x + \tau) = y$$

The task of robust certification is then to find methods that ensure the previous property. A key quantity in this case is the Lipschitz constant of the classifier.

2.1. Lipschitz property of Neural Networks

The Lipschitz constant has seen a growing interest in the last few years in the field of deep learning (Virmaux & Scaman, 2018; Fazlyab et al., 2019; Combettes & Pesquet, 2020; Béthune et al., 2021). Indeed, numerous results have shown that neural networks with a small Lipschitz constant exhibit better generalization (Bartlett et al., 2017), higher robustness to adversarial attacks (Szegedy et al., 2014; Farnia et al., 2019; Tsuzuku et al., 2018), better training stability (Xiao et al., 2018; Trockman et al., 2021), improved Generative Adversarial Networks (Arjovsky et al., 2017), etc. Formally, we define the Lipschitz constant with respect to the ℓ_2 norm of a Lipschitz continuous function f as follows:

$$\operatorname{Lip}_2(f) = \sup_{\substack{x, x' \in \mathcal{X} \\ x \neq x'}} \frac{\|f(x) - f(x')\|_2}{\|x - x'\|_2} \ .$$

Intuitively, if a classifier is Lipschitz, one can bound the impact of a given input variation on the output, hence obtaining guarantees on the adversarial robustness. We can formally characterize the robustness of a neural network with respect to its Lipschitz constant with the following proposition:

Proposition 1 (Tsuzuku et al. (2018))**.** *Let $\mathbf{f}$ be an L-Lipschitz continuous classifier for the ℓ_2 norm. Let $\varepsilon > 0$, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ the label of x. If at point x, the margin $\mathcal{M}_{\mathbf{f}}(x)$ satisfies:*

$$\mathcal{M}_{\mathbf{f}}(x) := \max(0, f_y(x) - \max_{y' \neq y} f_{y'}(x)) > \sqrt{2} L \varepsilon$$

then we have for every τ such that $\|\tau\|_2 \le \varepsilon$:

$$\operatorname*{argmax}_k f_k(x + \tau) = y$$

From Proposition 1, it is straightforward to compute a robustness certificate for a given point. Consequently, in order to build robust neural networks the margin needs to be large and the Lipschitz constant small to get optimal guarantees on the robustness for neural networks.

2.2. Certified Adversarial Robustness

Mainly two kinds of methods have been developed to come up with certified adversarial robustness. The first category relies on randomization and consists of convolving the input with a predefined probability distribution during both training and inference phases. Several works that rely on the method have proposed empirical (Cao & Gong, 2017; Liu et al., 2018; Pinot et al., 2019; 2020) and certified defenses (Lecuyer et al., 2018; Li et al., 2019a; Cohen et al., 2019; Salman et al., 2019; Yang et al., 2020). These methods are model-agnostic, in the sense they do not depend on the architecture of the classifier, and provide “high probability” certificates. In order to get non-vacuous provable guarantees, such approaches often require to query the network hundreds of times to infer the label of a single image. This computational cost naturally limits the use of these methods in practice.

The second approach directly exploits the Lipschitzness property with the design of built-in 1-Lipschitz layers. Contrarily to previous methods, these approaches provide deterministic guarantees. Following this line, one can either normalize the weight matrices by their largest singular values making the layer 1-Lipschitz, *e.g.* (Yoshida & Miyato, 2017; Miyato et al., 2018; Farnia et al., 2019; Anil et al., 2019) or project the weight matrices on the Stiefel manifold (Li et al., 2019b; Trockman et al., 2021; Singla & Feizi, 2021). The work of Li et al. (2019b), Trockman et al. (2021) and Singla & Feizi (2021) (denoted BCOP, Cayley and SOC respectively) are considered the most relevant approach to our work. Indeed, their approaches consist of projecting the weights matrices onto an orthogonal space in order to preserve gradient norms and enhance adversarial robustness by guaranteeing low Lipschitz constants. While both works have similar objectives, their execution is different. The BCOP layer (Block Convolution Orthogonal Parameterization) uses an iterative algorithm proposed by Björck et al. (1971) to orthogonalize the linear transform performed by a convolution. The SOC layer (Skew Orthogonal Convolutions) uses the property that if A is a skew symmetric matrix then $Q = \exp A$ is an orthogonal matrix. To approximate the exponential, the authors proposed to use a finite number of terms in its Taylor series expansion. Finally, the method proposed by Trockman et al. (2021) use the Cayley transform to orthogonalize the weights matrices. Given a skew symmetric matrix A, the Cayley transform consists in computing the orthogonal matrix $Q = (I - A)^{-1}(I + A)$. Both methods are well adapted to convolutional layers and are able to reach high accuracy levels on CIFAR datasets. Also, several works (Anil et al., 2019; Singla et al., 2021a; Huang et al., 2021b) proposed methods leveraging the properties of activation functions to constraints the Lipschitz of Neural Networks. These works are usually useful to help improving the performance of linear orthogonal layers.

2.3. Residual Networks

To prevent from gradient vanishing issues in neural networks during the training phase (Hochreiter et al., 2001), (He et al., 2016) proposed the Residual Network (ResNet) architecture. Based on this architecture, several works (Haber et al., 2017; E, 2017; Lu et al., 2018; Chen et al., 2018) proposed a “continuous time” interpretation inspired by dynamical systems that can be defined as follows.

Definition 2. *Let $(F_t)_{t\in[0,T]}$ be a family of functions on $\mathbb{R}^d$, we define the continuous time Residual Networks flow associated with F_t as:*

$$\begin{cases} x_0 &= x \in \mathcal{X} \\ \frac{dx_t}{dt} &= F_t(x_t) \text{ for } t \in [0,T] \end{cases}$$

This continuous time interpretation helps as it allows us to consider the stability of the forward propagation through the stability of the associated dynamical system. A dynamical system is said to be *stable* if two trajectories starting from an input and another one remain sufficiently close to each other all along the propagation. This stability property takes all its sense in the context of adversarial classification.

It was argued by Haber et al. (2017) that when F_t does not depend on t or vary slowly with time[1], the stability can be characterized by the eigenvalues of the Jacobian matrix $\nabla_x F_t(x_t)$: the dynamical system is stable if the real part of the eigenvalues of the Jacobian stay negative throughout the propagation. This property however only relies on intuition and this condition might be difficult to verify in practice. In the following, in order to derive stability properties, we study gradient flows and convex potentials, which are subclasses of Residual networks.

Other works (Huang et al., 2020b; Li et al., 2020) also proposed to enhance adversarial robustness using dynamical systems interpretations of Residual Networks. Both works argues that using particular discretization scheme would make gradient attacks more difficult to compute due to numerical stability. These works did not provide any provable guarantees for such approaches.

3. A Framework to design Lipschitz Layers

The continuous time interpretation of Definition 2 allows us to better investigate the robustness properties and assess how a difference of the initial values (the inputs) impacts the inference flow of the model. Let us consider two continuous flows x_t and z_t associated with F_t but differing in their respective initial values x_0 and z_0. Our goal is to characterize the time evolution of $\|x_t - z_t\|$ by studying its time derivative. We recall that every matrix $M \in \mathbb{R}^{d\times d}$ can be uniquely decomposed as the sum of a symmetric and skew-symmetric matrix $M = S(M) + A(M)$. By applying this decomposition to the Jacobian matrix $\nabla_x F_t(x)$ of F_t, we can show that the time derivative of $\|x_t - z_t\|^2$ only involves the symmetric part $S(\nabla_x F_t(x))$ (see Appendix B.1 for details).

For two symmetric matrices $S_1, S_2 \in \mathbb{R}^{d\times d}$, we denote $S_1 \preceq S_2$ if, for all $x \in \mathbb{R}^d$, $\langle x, (S_2 - S_1)x\rangle \geq 0$. By focusing on the symmetric part of the Jacobian matrix we can show in Appendix B.1 the following proposition.

Proposition 2. *Let $(F_t)_{t\in[0,T]}$ be a family of differentiable functions almost everywhere on $\mathbb{R}^d$. Let us assume that there exists two measurable functions $t \mapsto \mu_t$ and $t \mapsto \lambda_t$ such that*

$$\mu_t I \preceq S(\nabla_x F_t(x)) \preceq \lambda_t I$$

for all $x \in \mathbb{R}^d$, and $t \in [0,T]$. Then the flow associated with F_t satisfies for all initial conditions x_0 and z_0:

$$\|x_0 - z_0\| e^{\int_0^t \mu_s ds} \leq \|x_t - z_t\| \leq \|x_0 - z_0\| e^{\int_0^t \lambda_s ds}$$

The symmetric part plays even a more important role since one can show that a twice-differentiable function whose

[1]This blurry definition of "vary slowly" makes the property difficult to apply.

Jacobian is always skew-symmetric is actually linear (see Appendix C.1 for more details). However, constraining $S(\nabla_x F_t(x))$ in the general case is technically difficult and a solution resorts to a more intuitive parametrization of F_t as the sum of two functions $F_{1,t}$ and $F_{2,t}$ whose Jacobian matrix are respectively symmetric and skew-symmetric. Thus, such a parametrization enforces $F_{2,t}$ to be linear and skew-symmetric. For the choice of $F_{1,t}$, we propose to rely on potential functions: a function $F_{1,t}:\mathbb{R}^d\to\mathbb{R}^d$ derives from a simpler family of scalar valued function in $\mathbb{R}^d$, called the *potential*, via the gradient operation. Moreover, since the Hessian of the potential is symmetric, the Jacobian for $F_{1,t}$ is then also symmetric. If we add the convex property to this potential, its Hessian has positive eigenvalues. Therefore we introduce the following corollary. See proof in Appendix B.2

Corollary 1. *Let $(f_t)_{t\in[0,T]}$ be a family of convex differentiable functions on $\mathbb{R}^d$ and $(A_t)_{t\in[0,T]}$ a family of skew symmetric matrices. Let us define*

$$F_t(x) = -\nabla_x f_t(x) + A_t x,$$

then the flow associated with F_t satisfies for all initial conditions x_0 and z_0:

$$\|x_t - z_t\| \le \|x_0 - z_0\|$$

This simple property suggests that if we could parameterize F_t with convex potentials, it would be less sensitive to input perturbations and therefore more robust to adversarial examples. We also remark that the skew symmetric part is then norm-preserving. However, the discretization of such flow is challenging in order to maintain this property of stability. Note the question of the expressiveness of continuous and discretized flow induced by this decomposition is an open question.There is indeed no reason that this decomposition is fully expressive. This question is left as further work.

3.1. Discretized Flows

To study the discretization of the previous flow, let $t = 1,\dots,T$ be the discretized time steps and from now we consider the flow defined by $F_t(x) = -\nabla f_t(x) + A_t x$, with $(f_t)_{t=1,\dots,T}$ a family of convex differentiable functions on $\mathbb{R}^d$ and $(A_t)_{t=1,\dots,T}$ a family of skew symmetric matrices. The most basic method is the explicit Euler scheme as defined by:

$$x_{t+1} = x_t + F_t(x_t)$$

However, if $A_t \neq 0$, this discretized system might not satisfy $\|x_t - z_t\| \le \|x_0 - z_0\|$. Indeed, consider the simple example where $f_t = 0$. We then have:

$$\|x_{t+1} - z_{t+1}\|^2 - \|x_t - z_t\|^2 = \|A_t\,(x_t - z_t)\|^2.$$

Thus explicit Euler scheme cannot guarantee Lipschitzness when $A_t \neq 0$. To overcome this difficulty, the discretization step can be split in two parts, one for $\nabla_x f_t$ and one for A_t:

$$\begin{cases} x_{t+\frac{1}{2}} & = \textsc{step1}(x_t, \nabla_x f_t) \\ x_{t+1} & = \textsc{step2}(x_{t+\frac{1}{2}}, A_t) \end{cases}$$

This type of discretization scheme can be found for instance from Proximal Gradient methods where one step is explicit and the other is implicit. Then, we dissociate the Lipschitzness study of both terms of the flow.

3.2. Discretization scheme for $\nabla_x f_t$

To apply the explicit Euler scheme to $\nabla_x f_t$, an additional smoothness property on the potential functions is required to generalize the Lipschitzness guarantee to the discretized flows. Recall that a function f is said to be *L-smooth* if it is differentiable and if $x \mapsto \nabla_x f(x)$ is L-Lipschitz.

Proposition 3. *Let $t \in \{1,\cdots,T\}$ Let us assume that f_t is L_t-smooth. We define the following discretized ResNet gradient flow using h_t as a step size:*

$$x_{t+\frac{1}{2}} = x_t - h_t \nabla_x f_t(x_t)$$

Consider now two trajectories x_t and z_t with initial points $x_0 = x$ and $z_0 = z$ respectively, if $0 \le h_t \le \frac{2}{L_t}$, then

$$\|x_{t+\frac{1}{2}} - z_{t+\frac{1}{2}}\|_2 \le \|x_t - z_t\|_2$$

In Section 4, we describe how to parametrize a neural network layer to implement such a discretization step by leveraging the recent work on Input Convex Neural Networks (Amos et al., 2017).

Remark 1. *Another solution relies on the implicit Euler scheme: $x_{t+\frac{1}{2}} = x_t - \nabla_x f_t(x_{t+\frac{1}{2}})$. We show in Appendix C.2 that this strategy defines a 1-Lipschitz flow without further assumption on f_t than convexity. We propose an implementation. However preliminary experiments did not show competitive results and the training time is prohibitive. We leave this solution for future work.*

3.3. Discretization scheme for A_t

The second step of discretization involves the term with skew-symmetric matrix A_t. As mentioned earlier, the challenge is that the *explicit Euler discretization* is not contractive. More precisely, the following property

$$\|x_{t+1} - z_{t+1}\| \ge \|x_{t+\frac{1}{2}} - z_{t+\frac{1}{2}}\|$$

is satisfied with equality only in the special and useless case of $x_{t+\frac{1}{2}} - z_{t+\frac{1}{2}} \in \ker(A_t)$. Moreover, the implicit Euler discretization induces an increasing norm and hence does not satisfy the desired property of norm preservation neither.

Midpoint Euler method. We thus propose to use *Midpoint Euler* method, defined as follows:

$$x_{t+1} = x_{t+\frac{1}{2}} + A_t \frac{x_{t+1} + x_{t+\frac{1}{2}}}{2}$$
$$\iff x_{t+1} = \left(I - \frac{A_t}{2}\right)^{-1} \left(I + \frac{A_t}{2}\right) x_{t+\frac{1}{2}}.$$

Since A_t is skew-symmetric, $I - \frac{A_t}{2}$ is invertible. This update corresponds to the Cayley Transform of $\frac{A_t}{2}$ that induces an orthogonal mapping. This kind of layers was introduced and extensively studied in Trockman et al. (2021).

Exact Flow. One can define the simple differential equation corresponding to the flow associated with A_t

$$\frac{du_s}{ds} = A_t u_s, \quad u_0 = x_{t+\frac{1}{2}},$$

There exists an exact solution exists since A_t is linear. By taking the value at $s = \frac{1}{2}$, we obtained the following transformation:

$$x_{t+1} := u_{\frac{1}{2}} = e^{\frac{A}{2}} x_{t+\frac{1}{2}}.$$

This step is therefore clearly norm preserving but the matrix exponentiation is challenging and it requires efficient approximations. This trend was recently investigated under the name of Skew Orthogonal Convolution (SOC) (Singla & Feizi, 2021).

4. Parametrizing Convex Potentials Layers

As presented in the previous section, parametrizing the skew symmetric updates has been extensively studied by Trockman et al. (2021); Singla & Feizi (2021). In this paper we focus on the parametrization of symmetric update with the convex potentials proposed in 3. For that purpose, the Input Convex Neural Network (ICNN) (Amos et al., 2017) provide a relevant starting point that we will extend.

4.1. Gradient of ICNN

We use 1-layer ICNN (Amos et al., 2017) to define an efficient computation of Convex Potentials Flows. For any vectors $w_1, \ldots w_k \in \mathbb{R}^d$, and bias terms $b_1, \ldots, b_k \in \mathbb{R}$, and for ϕ a convex function, the potential F defined as:

$$F_{w,b} : x \in \mathbb{R}^d \mapsto \sum_{i=1}^{k} \phi(w_i^\top x + b_i)$$

defines a convex function in x as the composition of a linear and a convex function. Its gradient with respect to its input x is then:

$$x \mapsto \sum_{i=1}^{k} w_i \phi'(w_i^\top x + b_i) = \mathbf{W}^\top \phi'(\mathbf{W}x + \mathbf{b})$$

with $\mathbf{W} \in \mathbb{R}^{k \times d}$ and $\mathbf{b} \in \mathbb{R}^k$ are respectively the matrix and vector obtained by the concatenation of, respectively, $w_i^\top$ and b_i, and ϕ' is applied element-wise. Moreover, assuming ϕ' is L-Lipschitz, we have that $F_{w,b}$ is $L\|\mathbf{W}\|_2^2$-smooth. $\|\mathbf{W}\|_2$ denotes the spectral norm of $\mathbf{W}$, *i.e.*, the greatest singular value of $\mathbf{W}$ defined as:

$$\|\mathbf{W}\|_2 := \max_{x \neq 0} \frac{\|\mathbf{W}x\|_2}{\|x\|_2}$$

The reciprocal also holds: if $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-decreasing L-Lipschitz function, $\mathbf{W} \in \mathbb{R}^{k \times d}$ and $b \in \mathbb{R}^k$, there exists a convex $L\|\mathbf{W}\|_2^2$-smooth function $F_{w,b}$ such that

$$\nabla_x F_{w,b}(x) = \mathbf{W}^\top \sigma(\mathbf{W}x + \mathbf{b}),$$

where σ is applied element-wise. The next section shows how this property can be used to implement the building block and training of such layers.

4.2. Convex Potential layers

From the previous section, we derive the following *Convex Potential Layer*:

$$z = x - \frac{2}{\|\mathbf{W}\|_2^2} \mathbf{W}^\top \sigma(\mathbf{W}x + b)$$

Written in a matrix form, this layer can be implemented with every linear operation $\mathbf{W}$. In the context of image classification, it is beneficial to use convolutions[2] instead of generic linear transforms represented by a dense matrix.

Remark 2. *When* $\mathbf{W} \in \mathbb{R}^{1 \times d}$, $b = 0$ *and* $\sigma = \text{ReLU}$, *the* Convex Potential Layer *is equivalent to the HouseHolder activation function introduced in Singla et al. (2021a).*

Residual Networks (He et al., 2016) are also composed of other types of layers which increase or decrease the dimensionality of the flow. Typically, in a classical setting, the number of input channels is gradually increased, while the size of the image is reduced with pooling layers. In order to build a 1-Lipschitz Residual Network, all operations need to be properly scale or normalize in order to maintain the Lipschitz constant.

Increasing dimensionsionality. To increase the number of channels in a convolutional Convex Potential Layer, a zero-padding operation can be easily performed: an input x of size $c \times h \times w$ can be extended to some x' of size $c' \times h \times w$, where $c' > c$, which equals x on the c first channels and 0 on the $c' - c$ other channels.

[2]For instance, one can leverage the `Conv2D` and `Conv2D_transpose` functions of the PyTorch framework (Paszke et al., 2019)

Algorithm 1 Computation of a Convex Potential Layer

```
Require: Input: $x$, vector: $u$, weights: $\mathbf{W}, b$
Ensure: Compute the layer $z$ and return $u$
$v \leftarrow \mathbf{W}u / \|\mathbf{W}u\|_2$                                  } 1 iter. for training
$u \leftarrow \mathbf{W}^\top v / \|\mathbf{W}^\top v\|_2$                      } 100 iter. for inference
$h \leftarrow 2 / \left(\sum_i (\mathbf{W}u \cdot v)_i\right)^2$                }
return $x - h\left[\mathbf{W}^\top \sigma(\mathbf{W}x + b)\right], u$
```

Reducing dimensionsionality. Dimensionality reduction is another essential operation in neural networks. On one hand, its goal is to reduce the number of parameters and thus the amount of computation required to build the network. On the other hand it allows the model to progressively map the input space on the output dimension, which corresponds in many cases to the number of different labels K. In this context, several operations exist: pooling layers are used to extract information present in a region of the feature map generated by a convolution layer. One can easily adapt pooling layers (*e.g.* max and average) to make them 1-Lipschitz (Bartlett et al., 2017). Finally, a simple method to reduce the dimension is the product with a non-square matrix. In this paper, we simply implement it as the truncation of the output. This obviously maintains the Lipschitz constant.

4.3. Computing spectral norms

Our Convex Potential Layer, described in Equation 4.2, can be adapted to any kind of linear transformations (*i.e.* Dense or Convolutional) but requires the computation of the spectral norm for these transformations. Given that computation of the spectral norm of a linear operator is known to be NP-hard (Steinberg, 2005), an efficient approximate method is required during training to keep the complexity tractable.

Many techniques exist to approximate the spectral norm (or the largest singular value), and most of them exhibit a trade-off between efficiency and accuracy. Several methods exploit the structure of convolutional layers to build an upper bound on the spectral norm of the linear transform performed by the convolution (Jia et al., 2017; Singla et al., 2021b; Araujo et al., 2021). While these methods are generally efficient, they can less relevant and adapted to certain settings. For instance in our context, using a loose upper bound of the spectral norm will hinder the expressive power of the layer and make it too contracting.

For these reasons we rely on the Power Iteration Method (PM). This method converges at a geometric rate towards the largest singular value of a matrix. More precisely the convergence rate for a given matrix $\mathbf{W}$ is $O((\frac{\lambda_2}{\lambda_1})^k)$ after k iterations, independently from the choice for the starting vector, where $\lambda_1 > \lambda_2$ are the two largest singular values of $\mathbf{W}$. While it can appear to be computationally expensive due

#	S	M	L	XL
Conv. Layers	20	30	50	70
Channels	45	60	90	120
Lin. Layers	7	10	10	15
Lin. Features	2048	2048	4096	4096

Table 1. Architectures description for our Convex Potential Layers (CPL) neural networks with different capacities. We vary the number of Convolutional Convex Potential Layers, the number of Linear Convex Potential Layers, the number of channels in the convolutional layers and the width of fully connected layers. In the paper, they will be reported respectively as CPL-S, CPL-M, CPL-L and CPL-XL.

to the large number of required iterations for convergence, it is possible to drastically reduce the number of iterations during training. Indeed, as in (Miyato et al., 2018), by considering that the weights' matrices $\mathbf{W}$ change slowly during training, one can perform only one iteration of the PM for each step of the training and let the algorithm slowly converges along with the training process[3]. We describe with more details in Algorithm 1, the operations performed during a forward pass with a Convex Potential Layer. In order to keep the computation tractable in terms of time and memory, u and v are detached from graph computation for backpropagation.

However for evaluation purpose, we need to compute the certified adversarial robustness, and this requires to ensure the convergence of the PM. Therefore, we perform 100 iterations for each layer[4] at inference time. Also note that at inference time, the computation of the spectral norm only needs to be performed once for each layer.

5. Experiments

To evaluate our new 1-Lipschitz Convex Potential Layers, we carry out an extensive set of experiments. In this section, we first describe the details of our experimental setup. We then recall the concurrent approaches that build 1-Lipschitz Neural Networks and stress their limitations. Our experimental results are finally summarized in Section 5.1. By computing the certified and empirical adversarial accuracy of our networks on CIFAR10 and CIFAR100 classification tasks (Krizhevsky et al., 2009), we show that our architecture is competitive with state-of-the-art methods (Sections 5.3). In Appendix D, we also study the influence of some hyperparameters and demonstrate the stability and the scalability of our approach by training very deep neural

[3]Note that a typical training requires approximately 200K steps where 100 steps of PM is usually enough for convergence

[4]100 iterations of Power Method is sufficient to converge with a geometric rate.

networks up to 1000 layers without normalization tricks or gradient clipping. In Appendix D.6, we experimented different activation functions for our CPL layers, it turns out that the ReLU activation $\max(0, x)$ is the best performing one. In the following, all reported CPL models use ReLU activation functions.

In this section, we only stacked CPL layers: we do not alternate CPL layers and convolution layers. In Appendix D.3, we provide experiments on alternation of layers but the performances are quite poor. In our opinion, there are two main reasons for that. (1) The setting used in SOC (optimizer, losses, etc.) is different from the one used in CPL layers. We tried to fit CPL (resp. SOC) with their setting (resp. ours), but the performances were weaker. Alternating SOC layers and CPL layers did not improve the performance. (2) Even though Cayley has the same setting as ours, the regime of high performance is different for Orthogonal Layers (SOC and Cayley) and CPL layers. Small CPL nets are very fast to train but the performance is quite low in comparison with Orthogonal nets. When increasing the number of layers, Orthogonal Layers performances saturate and even decrease for a large number of layers, while the performance of CPL nets increases. Moreover, the training time of Deep Orthogonal nets is often prohibitive.

5.1. Training and Architectural Details

We demonstrate the effectiveness of our approach on a classification task with CIFAR10 and CIFAR100 datasets (Krizhevsky et al., 2009). We use a similar training configuration to the one proposed in (Trockman et al., 2021) We trained our networks with a batch size of 256 over 200 epochs. We use standard data augmentation (*i.e.*, random cropping and flipping), a learning rate of 0.001 with Adam optimizer (Diederik P. Kingma, 2014) without weight decay and a piecewise triangular learning rate scheduler. We used a margin loss[5] with margin parameter set to 0.7. Note that the margin parameter is a crucial parameter as described in Appendix D.5.

As other usual convolutional neural networks, we first stack few Convolutional CPLs and then stack some Linear CPLs for classification tasks. To validate the performance and the scalability of our layers, we will evaluate four different variations of different hyperparameters as described in Table 1, respectively named CPL-S, CPL-M, CPL-L and CPL-XL, ranked according to the number of parameters they have. In all our experiments, we made 3 independent trainings to evaluate accurately the models. All reported results are the average of these 3 runs.

[5]This procedure is also referred as Lipschitz Margin Training.

5.2. Concurrent Approaches

We compare our networks with SOC (Singla & Feizi, 2021) and Cayley (Trockman et al., 2021) networks which are to our knowledge the best performing approaches for deterministic 1-Lipschitz Neural Networks. Since our layers are fundamentally different from these ones, we cannot compare with the same architectures. We reproduced SOC results for with 10 and 20 layers, that we call respectively SOC-10 and SOC-20 in the same training setting, *i.e.* normalized inputs, cross entropy loss, SGD optimizer with learning rate 0.1 and multi-step learning rate scheduler. For Cayley layers networks, we reproduced their best reported model, *i.e.* KWLarge with width factor of 3.

The work of Singla et al. (2021a) propose three methods to improve certifiable accuracies from SOC layers: a new HouseHolder activation function (HH), last layer normalization (LLN), and certificate regularization (CR). The code associated with this paper is not open-sourced yet, so we just reported the results from their paper in ours results (Tables 2 and 3) under the name SOC+. We were being able to implement the LLN method in all models. This method largely improve the result of all methods on CIFAR100, so we used it for all networks we compared on CIFAR100 (ours and concurrent approaches).

5.3. Results

In this section, we present our results on adversarial robustness. We provide results on provable ℓ_2 robustness as well as empirical robustness on CIFAR10 and CIFAR100 datasets for all our models and the concurrent ones

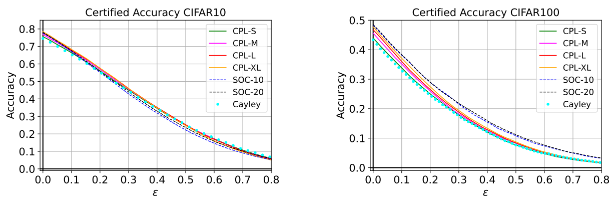


Figure 1. Certifiably robust accuracy in function of the perturbation ε for our CPL networks and its concurrent approaches (SOC and Cayley models) on CIFAR10 and CIFAR100 datasets.

Certified Adversarial Robustness. Results on CIFAR10 and CIFAR100 dataset are reported respectively in Tables 2 and 3. We also plotted certified accuracy in function of ε on Figure 1. On CIFAR10, our method outperforms the concurrent approaches in terms of standard and certified accuracies for every level of ε except SOC+ that uses additional tricks we did not use. On CIFAR100, our method performs slightly under the SOC networks but better than Cayley networks. Overall, our methods reach competitive results with SOC and Cayley layers.

	Clean Accuracy	Provable Accuracy (ε)			Time per epoch (s)
		36/255	72/255	108/255	
CPL-S	75.6	62.3	46.9	32.2	21.9
CPL-M	76.8	63.3	47.5	32.5	40.0
CPL-L	77.7	63.9	48.1	32.9	93.4
CPL-XL	78.5	64.4	48.0	33.0	163
Cayley (KW3)	74.6	61.4	46.4	32.1	30.8
SOC-10	77.6	62.0	45.0	29.5	33.4
SOC-20	78.0	62.7	46.0	30.3	52.2
SOC+-10	76.2	62.6	47.7	34.2	N/A
SOC+-20	76.3	62.6	48.7	36.0	N/A

Table 2. Results on the CIFAR10 dataset on standard and provably certifiable accuracies for different values of perturbations ε on CPL (ours), SOC and Cayley models. The average time per epoch in seconds is also reported in the last column. None of these networks uses Last Layer Normalization.

	Clean Accuracy	Provable Accuracy (ε)			Time per epoch (s)
		36/255	72/255	108/255	
CPL-S	44.0	29.9	19.1	11.0	22.4
CPL-M	45.6	31.1	19.3	11.3	40.7
CPL-L	46.7	31.8	20.1	11.7	93.8
CPL-XL	47.8	33.4	20.9	12.6	164
Cayley (KW3)	43.3	29.2	18.8	11.0	31.3
SOC-10	48.2	34.3	22.7	14.0	33.8
SOC-20	48.3	34.4	22.7	14.2	52.7
SOC+-10	47.1	34.5	23.5	15.7	N/A
SOC+-20	47.8	34.8	23.7	15.8	N/A

Table 3. Results on the CIFAR100 dataset on standard and provably certifiable accuracies for different values of perturbations ε on CPL (ours), SOC and Cayley models. The average time per epoch in seconds is also reported in the last column. All the reported networks use Last Layer Normalization.

Note that we observe a small gain using larger and deeper architectures for our models. This gain is less important as ε increases but the gain is non negligible for standard accuracies. In term of training time, our small architecture (CPL-S) trains very fast compared to other methods, while larger ones are longer to train.

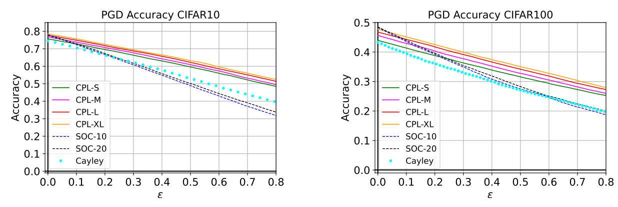


Figure 2. Accuracy against PGD attack with 10 iterations in function of the perturbation ε for our CPL networks and its concurrent approaches on CIFAR10 and CIFAR100 datasets.

Empirical Adversarial Robustness. We also reported in Figure 2 the accuracy of all the models against PGD ℓ_2-attack (Kurakin et al., 2016; Madry et al., 2018) for various levels of ϵ. We used 10 iterations for this attack. We remark here that our methods brings a large gain of robust accuracy over all other methods. On CIFAR10 for $\varepsilon = 0.8$, the gain of CPL-S over SOC-10 approach is more than 10%. For CIFAR100, the gain is about 10% too for $\varepsilon = 0.6$. We remark that using larger architectures lead in a more substantial gain in empirical robustness.

Our layers only provide an upper bound on the Lipschitz constant, while orthonormal layers as Cayley and SOC are built to exactly preserve the norms. This might negatively influence the certified accuracy since the effective Lipschitz constant is smaller than the theoretical one, hence leading to suboptimal certificates. This might explain why our method performs so well of empirical robustness task.

6. Conclusion

In this paper, we presented a new generic method to build 1-Lipschitz layers. We leverage the continuous time dynamical system interpretation of Residual Networks and show that using convex potential flows naturally defines 1-Lipschitz neural networks. After proposing a parametrization based on Input Convex Neural Networks (Amos et al., 2017), we show that our models reach competitive results in classification and robustness in comparison which other existing 1-Lipschitz approaches. We also experimentally show that our layers provide scalable approaches without further regularization tricks to train very deep architectures.

Exploiting the ResNet architecture for devising flows have been an important research topic. For example, in the context of generative modeling, Invertible Neural Networks (Behrmann et al., 2019) and Normalizing Flows (Rezende & Mohamed, 2015; Verine et al., 2021) are both import research topic. More recently, Sylvester Normalizing Flows (van den Berg et al., 2018) or Convex Potential Flows (Huang et al., 2021a) have had similar ideas to this present work but for a very different setting and applications. In particular, they did not have interest in the contraction property of convex flows and the link with adversarial robustness have been under-exploited.

Further work. Propoisition 2 suggests to constraint the symmetric part of the Jacobian of F_t. We proposed to decompose F_t as a sum of potential gradient and skew symmetric matrix. Finding other parametrizations is an open challenge. Our models may not express all 1-Lipschitz functions. Knowing which functions can be approximated by our CPL layers is difficult even in the linear case (see Appendix C.3). This is an important question that requires further investigation. One can also think of extending our work by the study of other dynamical systems. Recent architectures such as Hamiltonian Networks (Greydanus et al., 2019) and Momentum Networks (Sander et al., 2021) exhibit interesting properties. Finally, we hope to use similar approaches to build robust Recurrent Neural Networks (Sherstinsky, 2020) and Transformers (Vaswani et al., 2017).

References

Amos, B., Xu, L., and Kolter, J. Z. Input convex neural networks. In *International Conference on Machine Learning*, 2017.

Anil, C., Lucas, J., and Grosse, R. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, 2019.

Araujo, A., Negrevergne, B., Chevaleyre, Y., and Atif, J. On lipschitz regularization of convolutional layers using toeplitz matrix theory. *Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, 2017.

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

Béthune, L., González-Sanz, A., Mamalet, F., and Serrurier, M. The many faces of 1-lipschitz neural networks. *arXiv preprint arXiv:2104.05097*, 2021.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, 2013.

Björck, Å. et al. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM Journal on Numerical Analysis*, 1971.

Cao, X. and Gong, N. Z. Mitigating evasion attacks to deep neural networks via region-based classification. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pp. 278–287, 2017.

Carlini, N. et al. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, 2017.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, 2019.

Combettes, P. L. and Pesquet, J.-C. Lipschitz certificates for layered network structures driven by averaged activation operators. *SIAM Journal on Mathematics of Data Science*, 2020.

Croce, F. et al. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, 2020.

Diederik P. Kingma, J. B. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*, 2014.

E, W. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 2017.

Farnia, F., Zhang, J., and Tse, D. Generalizable adversarial training via spectral normalization. In *International Conference on Learning Representations*, 2019.

Fazlyab, M., Robey, A., Hassani, H., Morari, M., and Pappas, G. Efficient and accurate estimation of lipschitz constants for deep neural networks. In *Advances in Neural Information Processing Systems*, 2019.

Golub, G. H. et al. Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics*, 2000.

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

Gouk, H., Frank, E., Pfahringer, B., and Cree, M. J. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 2021.

Greydanus, S. J., Dzumba, M., and Yosinski, J. Hamiltonian neural networks. 2019.

Haber, E. et al. Stable architectures for deep neural networks. *Inverse problems*, 2017.

Hayou, S., Clerico, E., He, B., Deligiannidis, G., Doucet, A., and Rousseau, J. Stable resnet. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

Huang, C.-W., Chen, R. T. Q., Tsirigotis, C., and Courville, A. Convex potential flows: Universal probability distributions with optimal transport and convex optimization. In *International Conference on Learning Representations*, 2021a.

Huang, L., Liu, L., Zhu, F., Wan, D., Yuan, Z., Li, B., and Shao, L. Controllable orthogonalization in training dnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020a.

Huang, Y., Yu, Y., Zhang, H., Ma, Y., and Yao, Y. Adversarial robustness of stabilized neuralodes might be from obfuscated gradients. *Mathematical and Scientific Machine Learning*, 2020b.

Huang, Y., Zhang, H., Shi, Y., Kolter, J. Z., and Anandkumar, A. Training certifiably robust neural networks with efficient local lipschitz bounds. 2021b.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Jia, K., Tao, D., Gao, S., and Xu, X. Improving training of deep neural networks via singular value bounding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, 2018.

Li, B., Chen, C., Wang, W., and Carin, L. Certified adversarial robustness with additive noise. In *Advances in Neural Information Processing Systems*, 2019a.

Li, M., He, L., and Lin, Z. Implicit euler skip connections: Enhancing adversarial robustness via numerical stability. In *International Conference on Machine Learning*, pp. 5874–5883. PMLR, 2020.

Li, Q., Haque, S., Anil, C., Lucas, J., Grosse, R. B., and Jacobsen, J.-H. Preventing gradient attenuation in lipschitz constrained convolutional networks. In *Advances in Neural Information Processing Systems*. 2019b.

Liu, X., Cheng, M., Zhang, H., and Hsieh, C.-J. Towards robust neural networks via random self-ensemble. In

Proceedings of the European Conference on Computer Vision, 2018.

Lu, Y., Zhong, A., Li, Q., and Dong, B. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Moosavi-Dezfooli, S.-M., Fawzi, A., Uesato, J., and Frossard, P. Robustness via curvature regularization, and vice versa. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, 2013.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems* (NeurIPS), 2019.

Pinot, R., Meunier, L., Araujo, A., Kashima, H., Yger, F., Gouy-Pailler, C., and Atif, J. Theoretical evidence for adversarial robustness through randomization. In *Advances in Neural Information Processing Systems*, 2019.

Pinot, R., Ettedgui, R., Rizk, G., Chevaleyre, Y., and Atif, J. Randomization matters how to defend against strong adversarial attacks. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning*, 2015.

Salman, H., Li, J., Razenshteyn, I., Zhang, P., Zhang, H., Bubeck, S., and Yang, G. Provably robust deep learning via adversarially trained smoothed classifiers. In *Advances in Neural Information Processing Systems*, 2019.

Sander, M. E., Ablin, P., Blondel, M., and Peyré, G. Momentum residual neural networks. 2021.

Sedghi, H., Gupta, V., and Long, P. The singular values of convolutional layers. In *International Conference on Learning Representations*, 2018.

Sherstinsky, A. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

Singla, S. and Feizi, S. Skew orthogonal convolutions. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Singla, S., Singla, S., and Feizi, S. Householder activations for provable robustness against adversarial attacks. *arXiv preprint arXiv:2108.04062*, 2021a.

Singla, S. et al. Fantastic four: Differentiable and efficient bounds on singular values of convolution layers. In *International Conference on Learning Representations*, 2021b.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

Steinberg, D. Computation of matrix norms with applications to robust optimization. *Research thesis, Technion-Israel University of Technology*, 2005.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Trockman, A. et al. Orthogonalizing convolutional layers with the cayley transform. In *International Conference on Learning Representations*, 2021.

Tsuzuku, Y., Sato, I., and Sugiyama, M. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *Advances in Neural Information Processing Systems*, 2018.

van den Berg, R., Hasenclever, L., Tomczak, J., and Welling, M. Sylvester normalizing flows for variational inference. In *proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Verine, A., Chevaleyre, Y., Rossi, F., and benjamin negrevergne. On the expressivity of bi-lipschitz normalizing flows. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021.

Virmaux, A. and Scaman, K. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*. 2018.

Wang, J., Chen, Y., Chakraborty, R., and Yu, S. X. Orthogonal convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Wong, E., Schmidt, F., Metzen, J. H., and Kolter, J. Z. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems*, 2018.

Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S., and Pennington, J. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, 2018.

Yang, G., Duan, T., Hu, J. E., Salman, H., Razenshteyn, I., and Li, J. Randomized smoothing of all shapes and sizes. In *International Conference on Machine Learning*, 2020.

Yoshida, Y. and Miyato, T. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

A. Further Related Work

Lipschitz Regularization for Robustness. Based on the insight that Lipschitz Neural Networks are more robust to adversarial attacks, researchers have developed several techniques to regularize and constrain the Lipschitz constant of neural networks. However the computation of the Lipschitz constant of neural networks has been shown to be NP-hard (Virmaux & Scaman, 2018). Most methods therefore tackle the problem by reducing or constraining the Lipschitz constant at the layer level. For instance, the work of Cisse et al. (2017); Huang et al. (2020a) and Wang et al. (2020) exploit the orthogonality of the weights matrices to build Lipschitz layers. Other approaches (Gouk et al., 2021; Jia et al., 2017; Sedghi et al., 2018; Singla et al., 2021b; Araujo et al., 2021) proposed to estimate or upper-bound the spectral norm of convolutional and dense layers using for instance the power iteration method (Golub et al., 2000). While these methods have shown interesting results in terms of accuracy, empirical robustness and efficiency, they can not provide provable guarantees since the Lipschitz constant of the trained networks remains unknown or vacuous.

Reshaped Kernel Methods. It has been shown by Cisse et al. (2017) and Tsuzuku et al. (2018) that the spectral norm of a convolution can be upper-bounded by the norm of a reshaped kernel matrix. Consequently, orthogonalizing directly this matrix upper-bound the spectral norm of the convolution by 1. While this method is more computationally efficient than orthogonalizing the whole convolution, it lacks expressivity as the other singular values of the convolution are certainly too constrained.

B. Proofs

B.1. Proof of Proposition 2

Proof. Consider the time derivative of the square difference between the two flows x_t and z_t associated with the function F_t and following the definition 2:

$$\begin{aligned}\frac{d}{dt}\|x_t - z_t\|_2^2 &= 2\big\langle x_t - z_t, \frac{d}{dt}(x_t - z_t)\big\rangle \\ &= 2\big\langle x_t - z_t, F_{\theta_t}(x_t) - F_{\theta_t}(z_t)\big\rangle \\ &= 2\big\langle x_t - z_t, \int_0^1 \nabla_x F_{\theta_t}(x_t + s(z_t - z_t))(x_t - z_t)ds\big\rangle, \text{ by Taylorn-Lagrange formula} \\ &= 2\int_0^1 \big\langle x_t - z_t, \nabla_x F_{\theta_t}(x_t + s(z_t - z_t))(x_t - z_t)\big\rangle ds \\ &= 2\int_0^1 \big\langle x_t - z_t, S(\nabla_x F_{\theta_t}(x_t + s(z_t - z_t)))(x_t - z_t)\big\rangle ds\end{aligned}$$

In the last step, we used that for every skew-symmetric matrix A and vector x, $\|x, Ax\| = 0$. Since $\mu_t I \preceq S(\nabla_x F_{\theta_t}(x_t + s(z_t - y_t))) \preceq \lambda_t I$, we get

$$2\mu_t\|x_t - z_t\|_2^2 \leq \frac{d}{dt}\|x_t - z_t\|_2^2 \leq 2\lambda_t\|x_t - z_t\|_2^2$$

Then by Gronwall Lemma, we have

$$\|x_0 - y_0\|e^{\int_0^t \mu_s ds} \leq \|x_t - y_t\| \leq \|x_0 - y_0\|e^{\int_0^t \lambda_s ds}$$

which concludes the proof. □

B.2. Proof of Corollary 1

Proof. For all t, x, we have $F_t(x) = -\nabla_x f_t(x) + A_t x$ so $\nabla_x F_t(x) = -\nabla_x^2 f_t(x) + A_t$. Then $S(\nabla_x F_t(x)) = -\nabla_x^2 f_t(x)$. Since f is convex, we have $\nabla_x^2 f_t(x) \succeq 0$. So by application of Proposition 2, we deduce $\|x_t - y_t\| \leq \|x_0 - y_0\|$ for all trajectories starting from x_0 and y_0. □

B.3. Proof of Proposition 3

Proof. With $c_t = \|x_t - z_t\|_2^2$, we can write:

$$c_{t+\frac{1}{2}} - c_t = -2h_t\big\langle x_t - z_t, \nabla_x F_{\theta_t}(x_t) - \nabla_x F_{\theta_t}(z_t)\big\rangle + h_t^2\|\nabla_x F_{\theta_t}(z_t) - \nabla_x F_{\theta_t}(z_t)\|_2^2$$

This equality allows us to derive the equivalence between $c_{t+1} \le c_t$ and:

$$\frac{h_t}{2}\|\nabla F_{\theta_t}(x_t) - \nabla F_{\theta_t}(z_t)\|_2^2 \le \langle x_t - z_t, \nabla F_{\theta_t}(z_t) - \nabla F_{\theta_t}(z_t)\rangle$$

Moreover, assuming that F_{θ_t} being that:

$$\frac{1}{L_t}\|\nabla_x F_{\theta_t}(x_t) - \nabla_x F_{\theta_t}(z_t)\|_2^2 \le \big\langle x_t - z_t, \nabla_x F_{\theta_t}(x_t) - \nabla_x F_{\theta_t}(z_t)\big\rangle$$

We can see with this last inequality that if we enforce $h_t \le \frac{2}{L_t}$, we get $c_{t+\frac{1}{2}} \le c_t$ which concludes the proof. □

C. Additional Results

C.1. Functions whose gradient is skew-symmetric everywhere

Let $F := (F_1, \dots, F_d) : \mathbb{R}^d \to \mathbb{R}^d$ be a twice differentiable function such that $\nabla F(x)$ is skew-symmetric for all $x \in \mathbb{R}^d$. Then we have for all i, j, k:

$$\partial_i\partial_j F_k = -\partial_i\partial_k F_j = -\partial_k\partial_i F_j = \partial_k\partial_j F_i = \partial_j\partial_k F_i = -\partial_j\partial_i F_k = -\partial_i\partial_j F_k$$

So we have $\partial_i\partial_j F_k = 0$ and then F is linear: there exists a skew-symmetric matrix A such that $F(x) = Ax$

C.2. Implicit discrete convex potential flows

Let us define the implicit update $x_{t+\frac{1}{2}} = x_t - \nabla_x f_t(x_{t+\frac{1}{2}})$. Let us remark that $x_{t+\frac{1}{2}}$ is uniquely defined as:

$$x_{t+\frac{1}{2}} = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2}\|x - x_t\|^2 + f_t(x)$$

We recognized here the proximal operator of f_t that is uniquely defined since f_t is convex. Moreover we have for two trajectories x_t and z_t:

$$\begin{aligned}\|x_t - z_t\|_2^2 &= \|x_{t+\frac{1}{2}} - z_{t+\frac{1}{2}} + \nabla_x f_t(x_{t+\frac{1}{2}}) - \nabla_x f_t(z_{t+\frac{1}{2}})\|_2^2 \\ &= \|x_{t+\frac{1}{2}} - z_{t+\frac{1}{2}}\|^2 + 2\langle x_t - z_t, \nabla_x f_t(x_{t+\frac{1}{2}}) - \nabla_x f_t(z_{t+\frac{1}{2}})\rangle + \|\nabla_x f_t(x_{t+\frac{1}{2}}) - \nabla_x f_t(z_{t+\frac{1}{2}})\|_2^2 \\ &\ge \|x_{t+\frac{1}{2}} - z_{t+\frac{1}{2}}\|^2\end{aligned}$$

where the last inequality is deduced from the convexity of f_t. So, without any further assumption on f_t, the discretized implicit convex potential flow is 1-Lipschitz.

To compute such a layer, one could solve the proximal operator strongly convex-minimization optimization problem. This strategy is not computationally efficient and not scalable.

C.3. Expressivity of discretized convex potential flows

Let us define $\mathcal{S}_1(\mathbb{R}^{d\times d})$ the space of real symmetric matrices with singular values bounded by 1. Let us also define $\mathcal{M}_1(\mathbb{R}^{d\times d})$ the space of real matrices with singular values bounded by 1 in absolute value. Let $\mathcal{P}(\mathbb{R}^{d\times d}) = \{A \in \mathbb{R}^{d\times d} | \exists n \in \mathbb{N}, S_1, \dots, S_n \in \mathcal{S}_1(\mathbb{R}^d \times d) \text{ s.t. } A = S_1 \dots S_n\}$. Then one can prove[6] that $\mathcal{P}(\mathbb{R}^{d\times d}) \ne \mathcal{M}_1(\mathbb{R}^{d\times d})$. Thus there exists $A \in \mathcal{M}_1(\mathbb{R}^{d\times d})$ such that for all matrices n, for all matrices $S_1, \dots, S_n \in \mathcal{S}_1(\mathbb{R}^{d\times d})$ such that $M \ne S_1, ..., S_n$.

Applied to the expressivity of discretized convex potential flows, the previous result means that there exists a 1-Lipschitz linear function that cannot be approximated as a discretized flow of any depth of convex linear 1-smooth potential flows as in Proposition 3. Indeed such a flow would write: $x \mapsto \prod_i (1 - 2S_i)x$ where S_i are symmetric matrices whose eigenvalues are in $[0, 1]$, in other words such transformations are exactly described by $x \mapsto Mx$ for some $M \in \mathcal{P}(\mathbb{R}^{d\times d})$.

[6]A proof and justification of this result can be found here: https://mathoverflow.net/questions/60174/factorization-of-a-real-matrix-into-hermitian-x-hermitian-is-it-stable

D. Additional experiments

D.1. Training stability: scaling up to 1000 layers

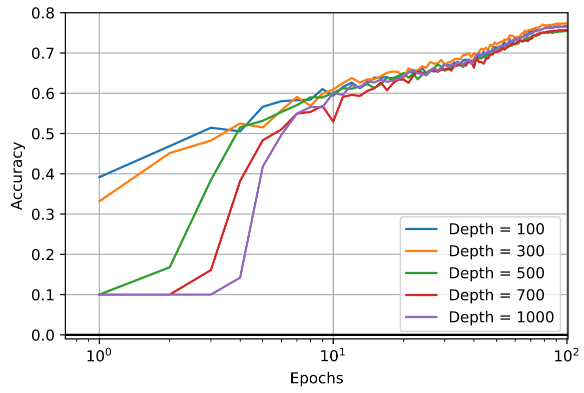


Figure 3. Standard test accuracy in function of the number of epochs (log-scale) for various depths for our neural networks $(100, 300, 500, 700, 1000)$.

While the Residual Network architecture limits, by design, gradient vanishing issues, it still suffers from exploding gradients in many cases (Hayou et al., 2021). To prevent such scenarii, batch normalization layers (Ioffe & Szegedy, 2015) are used in most Residual Networks to stabilize the training.

Recently, several works (Miyato et al., 2018; Farnia et al., 2019) have proposed to normalize the linear transformation of each layer by their spectral norm. Such a method would limit exploding gradients but would again suffer from gradient vanishing issues. Indeed, spectral normalization might be too restrictive: dividing by the spectral norm can make other singular values vanishingly small. While more computationally expensive (spectral normalization can be done with 1 Power Method iteration), orthogonal projections prevent both exploding and vanishing issues.

On the contrary the architecture proposed in this paper has the advantage to naturally control the gradient norm of the output with respect to a given layer. Therefore, our architecture can get the best of both worlds: limiting exploding and vanishing issues while maintaining scalability. To demonstrate the scalability of our approach, we experiment the ability to scale our architecture to very high depth (up to 1000 layers) without any additional normalization/regularization tricks, such as Dropout (Srivastava et al., 2014), Batch Normalization (Ioffe & Szegedy, 2015) or gradient clipping (Pascanu et al., 2013). With the work done by (Xiao et al., 2018), which leverage Dynamical Isometry and a Mean Field Theory to train a 10000 layers neural network, we believe, to the best of our knowledge, to be the second to perform such training. For sake of computation efficiency, we limit this experiment to architecture with 30 feature maps. We report the accuracy in terms of epochs for our architecture in Figure 3 for a varying number of convolutional layers. It is worth noting that for the deepest networks, it may take a few epochs before the start of convergence. As (Xiao et al., 2018), we remark there is no gain in using very deep architecture for this task.

D.2. Relaxing linear layers

	h = 1.0	**h = 0.1**	**h = 0.01**
Clean	85.10	82.23	78.53
PGD ($\varepsilon = 36/255$)	61.45	62.99	60.98

The table above shows the result of the relaxed training on the CIFAR10 dataset of our StableBlock architecture, i.e. we fixed the step h_t in the discretized convex potential flow of Proposition 3. Increasing the constant h allows for an important improvement in the clean accuracy, but we loose in robust empirical accuracy. While computing the certified accuracy is not possible in this case due to the unknown value of the Lipschitz constant, we can still notice that the training of the network are still stable without normalization tricks, and offer a non-negligible level of robustness.

D.3. Mixing CPL and Orthogonal Convolutions layers

	CPL&SOC-20	CPL&Cayley (KW3)	SOC-20	Cayley(KW3)
Clean	71.06	74.89	77.1	74.6
Provable ($\varepsilon = 36/255$)	53.03	61.69	62.5	61.4

The table above shows the result of the training on the CIFAR10 dataset of SOC-20 and KWLarge architecture mixed with Convex Potential Layers : respectively each orthogonal convolution, linear layer is followed respectively by a convolutional, linear CPL. We notice that mixing the layer is difficult to train and at stakes as we do not recover the best performance of SOC-20 and improves slightly the performance of KWLarge architectures.

D.4. Effect of Batch Size in Training

	Batch size	Clean Accuracy	Provable Accuracy (ε)			Time per epoch (s)
			36/255	72/255	108/255	
CPL-S	64	76.5	62.9	47.3	32.0	48
	128	76.1	62.8	47.1	32.3	31
	256	75.6	62.3	46.9	32.2	22
CPL-M	64	77.4	63.6	47.4	32.1	77
	128	77.2	63.5	47.5	32.1	50
	256	76.8	63.2	47.4	32.4	40
CPL-L	64	78.4	64.2	47.8	32.2	162
	128	78.2	64.3	47.9	32.5	109
	256	77.6	63.9	48.1	32.7	93
CPL-XL	64	78.9	64.2	47.2	31.2	271
	128	78.9	64.2	47.5	31.8	198
	256	78.5	64.4	47.8	32.4	163

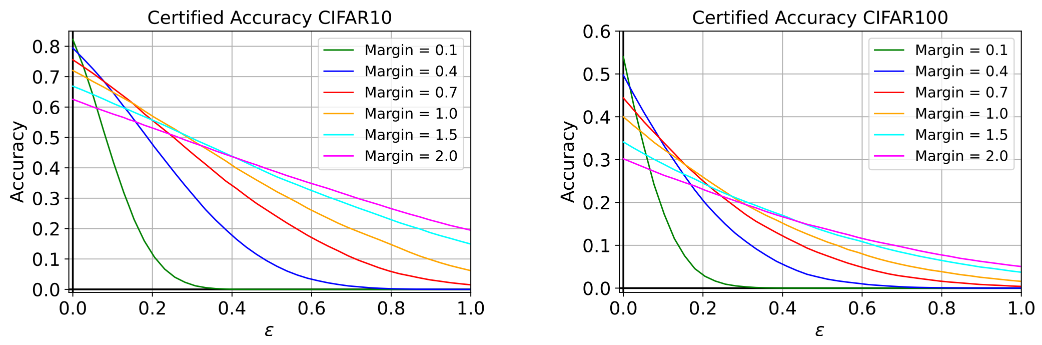
Table 4. Results on the CIFAR10 dataset on standard and provably certifiable accuracies for different values of perturbations ε on CPL (ours) models with various batch sizes. The average time per epoch in seconds is also reported in the last column. All the reported networks use Last Layer Normalization.

	Batch size	Clean Accuracy	Provable Accuracy (ε)			Time per epoch (s)
			36/255	72/255	108/255	
CPL-S	64	45.6	30.8	19.3	11.2	47
	128	44.9	30.7	19.2	11.0	31
	256	44.0	29.9	19.1	10.9	23
CPL-M	64	46.6	31.6	19.6	11.6	78
	128	46.3	31.1	19.7	11.5	55
	256	45.6	31.1	19.3	11.3	41
CPL-L	64	48.1	32.7	20.3	11.7	163
	128	47.4	32.3	20.0	11.8	116
	256	46.8	31.8	20.1	11.7	95
CPL-XL	64	49.0	33.7	21.1	12.0	293
	128	48.0	33.7	21.0	12.1	209
	256	47.8	33.4	20.9	12.6	164

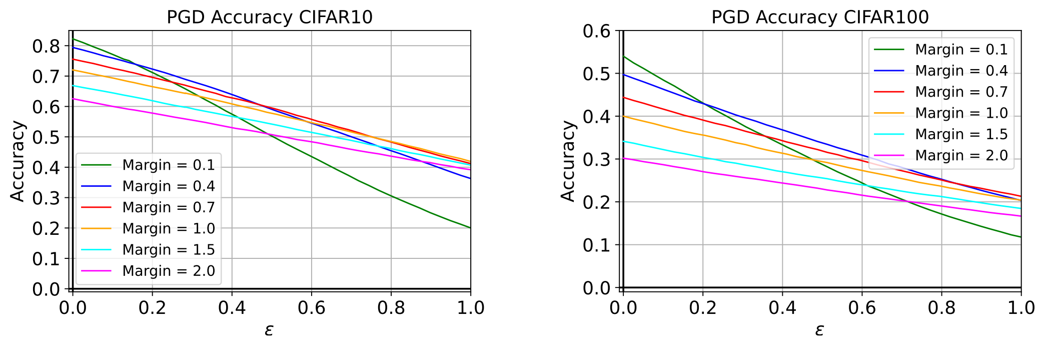
Table 5. Results on the CIFAR100 dataset on standard and provably certifiable accuracies for different values of perturbations ε on CPL (ours) models with various batch sizes. The average time per epoch in seconds is also reported in the last column. All the reported networks use Last Layer Normalization.

In Tables 4 and 5, we tried three different batch sizes (64, 128 and 256) for training our networks on CIFAR10 and CIFAR100 datasets, we remark a gain in standard accuracy in reducing the batch size for all settings. As the perturbation becomes larger, the gain in accuracy is reduced and can even in some cases we may loose some points in robustness.

D.5. Effect of the Margin Parameter



(a) Certifiably robust accuracy in function of the perturbation ε for our CPL-S network with different margin parameters on CIFAR10 and CIFAR100 datasets.



(b) Certifiably robust accuracy in function of the perturbation ε for our CPL-S network with different margin parameters on CIFAR10 and CIFAR100 datasets.

In these experiments we varied the margin parameter in the margin loss in Figures 4a and 4b. It clearly exhibits a tradeoff between standard and robust accuracy. When the margin is large, the standard accuracy is low, but the level of robustness remain high even for "large" perturbations. On the opposite, when the margin is small, we get a high standard accuracy but we are unable to keep a good robustness level as the perturbation increases. It is verified both on certified and empirical robustness.

D.6. Effect of Different Convex Activation: ReLU, Sigmoid, Tanh

Dataset	Activation	Acc.	Certified Acc.	AutoAttack	PGD Attack
CIFAR10	ReLU	**0.7682**	**0.6314**	**0.7129**	**0.7260**
	Tanh	0.5663	0.4223	0.5003	0.5145
	Sigmoid	0.5239	0.3741	0.4519	0.4715
CIFAR100	ReLU	**0.4546**	**0.3072**	**0.3910**	**0.4143**
	Tanh	0.2790	0.1766	0.2289	0.2477
	Sigmoid	0.2436	0.1568	0.2033	0.2200

Table 6. Results on the CIFAR10 and CIFAR100 dataset on standard and provably certifiable accuracies as well as Auto and PGD attack for different nonlinear activations.

In Tables 6, we tried three different nonlinear convex activation for our Convex Potential Layer: ReLU, Tanh and Sigmoid. We can observe that models trained with ReLU activation offer much higher accuracies than the other two non-linear activations.