Almost-Orthogonal Layers for Efficient General-Purpose Lipschitz Networks

Bernd Prach(✉) and Christoph H. Lampert

Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria
{bprach,chl}@ist.ac.at

Abstract. It is a highly desirable property for deep networks to be robust against small input changes. One popular way to achieve this property is by designing networks with a small Lipschitz constant. In this work, we propose a new technique for constructing such *Lipschitz networks* that has a number of desirable properties: it can be applied to any linear network layer (fully-connected or convolutional), it provides formal guarantees on the Lipschitz constant, it is easy to implement and efficient to run, and it can be combined with any training objective and optimization method. In fact, our technique is the first one in the literature that achieves all of these properties simultaneously.

Our main contribution is a rescaling-based weight matrix parametrization that guarantees each network layer to have a Lipschitz constant of at most 1 and results in the learned weight matrices to be close to orthogonal. Hence we call such layers *almost-orthogonal Lipschitz (AOL)*. Experiments and ablation studies in the context of image classification with certified robust accuracy confirm that AOL layers achieve results that are on par with most existing methods. Yet, they are simpler to implement and more broadly applicable, because they do not require computationally expensive matrix orthogonalization or inversion steps as part of the network architecture.

We provide code at https://github.com/berndprach/AOL.

Keywords: Lipschitz networks · Orthogonality · Robustness

1 Introduction

Deep networks are often the undisputed state of the art when it comes to solving computer vision tasks with high accuracy. However, the resulting systems tend to be not very *robust*, e.g., against small changes in the input data. This makes them untrustworthy for safety-critical high-stakes tasks, such as autonomous driving or medical diagnosis.

A typical example of this phenomenon are *adversarial examples* [20]: imperceptibly small changes to an image can drastically change the outputs of a

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-19803-8_21.

S. Avidan et al. (Eds.): ECCV 2022, LNCS 13681, pp. 350–365, 2022.
https://doi.org/10.1007/978-3-031-19803-8_21

deep learning classifier when chosen in an adversarial way. Since their discovery, numerous methods were developed to make networks more robust against adversarial examples. However, in response a comparable number of new attack forms were found, leading to an ongoing cat-and-mouse game. For surveys on the state of research, see, e.g., [6,17,26].

A more principled alternative is to create deep networks that are robust by design, for example, by restricting the class of functions they can represent. Specifically, if one can ensure that a network has a small *Lipschitz constant*, then one knows that small changes to the input data will not result in large changes to the output, even if the changes are chosen adversarially.

A number of methods for designing such *Lipschitz networks* have been proposed in the literature, which we discuss in Sect. 3. However, all of them have individual limitations. In this work, we introduce the AOL (for *almost-orthogonal Lipschitz*) method. It is the first method for constructing Lipschitz networks that simultaneously meets all of the following desirable criteria:

Generality. AOL is applicable to a wide range of network architectures, in particular most kinds of fully-connected and convolutional layers. In contrast, many recent methods work only for a restricted set of layer types, such as only fully-connected layers or only convolutional layers with non-overlapping receptive fields.

Formal Guarantees. AOL provably guarantees a Lipschitz constant 1. This is in contrast to methods that only encourage small Lipschitz constants, e.g., by regularization.

Efficiency. AOL causes only a small computational overhead at training time and none at all at prediction time. This is in contrast to methods that embed expensive iterative operations such as matrix orthogonalization or inversion steps into the network layers.

Modularity. AOL can be treated as a black-box module and combined with arbitrary training objective functions and optimizers. This is in contrast to methods that achieve the Lipschitz property only when combined with, e.g., specific loss-rescaling or projection steps during training.

AOL's name stems from the fact that the weight matrices it learns are approximately orthogonal. In contrast to prior work, this property is not enforced explicitly, which would incur a computational cost. Instead, almost-orthogonal weight matrices emerge organically during network training. The reason is that AOL's rescaling step relies on an upper bound to the Lipschitz constant that is tight for parameter matrices with orthogonal columns. During training, matrices without that property are put at the disadvantage of resulting in outputs of smaller dynamic range. As a consequence, orthogonal matrices are able to achieve smaller values of the loss and are therefore preferred by the optimizer.

2 Notation and Background

A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is called *L-Lipschitz continuous* with respect to norms $\|.\|_{\mathbb{R}^n}$ and $\|.\|_{\mathbb{R}^m}$, if it fulfills

$$\|f(x) - f(y)\|_{\mathbb{R}^m} \leq L\|x - y\|_{\mathbb{R}^n}, \tag{1}$$

for all x and y, where L is called the *Lipschitz constant.* In this work we only consider Lipschitz-continuity with respect to the Euclidean norm, $\|.\|_2$, and mainly for $L = 1$. For conciseness of notation, we refer to such 1-Lipschitz continuous functions simply as *Lipschitz functions.*

For any linear (actually affine) function f, the Lipschitz property can be verified by checking if the function's Jacobian matrix, J_f, has *spectral norm* $\|J_f\|_{\text{spec}}$ less or equal to 1, where

$$[J_f]_{ij} = \frac{\partial f_i}{\partial x_j} \qquad \text{and} \qquad \|M\|_{\text{spec}} = \max_{\|v\|_2=1} \|Mv\|_2. \tag{2}$$

The spectral norm of a matrix M can in fact be computed numerically as it is identical to the largest singular value of the matrix. This, however, typically requires iterative algorithms that are computationally expensive in high-dimensional settings. An exception is if M is an orthogonal matrix, i.e. $M^\top M = I$, for I the identity matrix. In that case we know that all its singular values are 1 and $\|Mv\|_2 = \|v\|_2$ for all v, so the corresponding linear transformation is Lipschitz.

Throughout this work we consider a deep neural network as a concatenation of linear layers (fully-connected or convolutional) alternating with non-linear activation functions. We then study the problem how to ensure that the resulting network function is Lipschitz.

It is known that computing the exact Lipschitz constant of a neural network is an NP-hard problem [24]. However, upper bounds can be computed more efficiently, e.g., by multiplying the individual Lipschitz constants of all layers.

3 Related Work

The first attempts to train deep networks with small Lipschitz constant used ad-hoc techniques, such as weight clipping [2] or regularizing either the network gradients [10] or the individual layers' spectral norms [14]. However, these techniques do not formally guarantee bounds on the Lipschitz constant of the trained network. Formal guarantees are provided by constructions that ensure that each individual network layer is Lipschitz. Combined with Lipschitz activation functions, such as ReLU, MaxMin or tanh, this ensures that the overall network function is Lipschitz.

In the following, we discuss a number of prior methods for obtaining Lipschitz networks. A structured overview of their properties can be found in Table 1.

Table 1. Overview of the properties of different methods for learning Lipschitz networks.Columns indicate: *E (efficiency)*: no internal iterative procedure required, scales well with the input size. *F (formal guarantees)*: provides a guarantee abound the Lipschitz constant of the trained network. *G (generality)*: can be applied to fully-connected as well as convolutional layers. *M (modularity)*: can be used with any training objective and optimization method. $\sim$ symbols indicate that a property is partially fulfilled. Superscripts provide further explanations: [1] requires a regularization loss. [2] internal methods would have to be run to convergence. [3] iterative procedure that can be split between training steps. [4] requires matrix orthogonalization. [5] requires inversion of an input-sized matrix. [6] requires circular padding and full-image kernel size. [7] requires large kernel sizes to ensure orthogonality

Method	E	F	G	M	Methodology
WGAN [2]	✓	✓	✓	✗	Weight clipping
WGAN-GP [10]	✓	✗	✓	$\sim$[1]	Regularization
Parseval Networks [7]	✓	✗	✓	$\sim$[1]	Regularization
OCNN [25]	✓	✗	✓	$\sim$[1]	Regularization
SN [14]	✗	$\sim$[2]	✓	✓	parameter rescaling
LCC [9]	✗	$\sim$[2]	✓	✗	Parameter rescaling
LMT [23]	$\sim$[3]	✗	✓	✗	Loss rescaling
GloRo [12]	$\sim$[3]	$\sim$[2]	✓	✗	Loss rescaling
BCOP [13]	✗[4]	✓	✓	✓	Explicit orthogonalization
GroupSort[1]	✗[4]	$\sim$[2]	✗	✓	Explicit orthogonalization
ONI [11]	✗	$\sim$[2]	✓	✓	Explicit orthogonalization
Cayley Convs [21]	$\sim$[5]	✓	$\sim$[6]	✓	Explicit orthogonalization
SOP [18]	✗	$\sim$[2]	$\sim$[7]	✓	Explicit orthogonalization
ECO [27]	✗[4]	✓	$\sim$[6]	✓	Explicit orthogonalization
AOL (proposed)	✓	✓	✓	✓	Parameter rescaling

Bound-Based Methods. The Lipschitz property of a network layer could be achieved trivially: one simply computes the layer's Lipschitz constant, or an upper bound, and divides the layer weights by that value. Applying such a step after training, however, does not lead to satisfactory results in practice, because the dynamic range of the network outputs is reduced by the product of the scale factors. This can be seen as a reduction of network capacity that prevents the network from fitting the training data well. Instead, it makes sense to incorporate the Lipschitz condition already at training time, such that the optimization can attempt to find weight matrices that lead to a network that is not only Lipschitz but also able to fit the data well.

The *Lipschitz Constant Constraint (LCC)* method [9] identifies all weight matrices with spectral norm above a threshold λ after each weight update and rescales those matrices to have a spectral norm exactly λ. *Lipschitz Margin Training (LMT)* [23] and *Globally-Robust Neural Networks (GloRo)* [12] approximate the overall Lipschitz constant from numeric estimates of the largest singular values of the layers' weight matrices. They integrate this value as a scale factor into their respective loss functions.

In the context of deep learning, controlling only the Lipschitz constant of each layer separately has some drawbacks. In particular, the product of the individual Lipschitz constants might grossly overestimate the network's actual Lipschitz constant. The reason is that the Lipschitz constant of a layer is determined by a single vector direction of maximal expansion. When concatenating multiple layers, their directions of maximal expansion will typically not be aligned, especially with in between nonlinear activations. As a consequence, the actual maximal amount of expansion will be smaller than the product of the per-layer maximal expansions. This causes the variance of the activations to shrink during the forward pass through the network, even though in principle a sequence of 1-Lipschitz operations could perfectly preserve it. Analogously, the magnitude of the gradient signal shrinks with each layer during the backwards pass of backpropagation training, which can lead to vanishing gradient problems.

Orthogonality-Based Method. A way to address the problems of variance-loss and vanishing gradients is to exploit *orthogonality*, which has been found useful in computer vision and machine learning [3,8,15]. Specifically one uses network layers that encode *orthogonal* linear operations. These are 1-Lipschitz, so the overall network will also have that property. However, they are also *isotropic*, in the sense that they preserve data variance and gradient magnitude in all directions, not just a single one.

For fully-connected layers, it suffices to ensure that the weight matrices themselves are orthogonal. The *GroupSort* [1] architecture achieves this using classic results from numeric analysis [4]. The authors parameterize an orthogonal weight matrix as a specific matrix power series, which they embed in truncated form into the network architecture. *Orthogonalization by Newton's Iterations (ONI)* [11] parameterizes orthogonal weight matrices as $\left(VV^\top\right)^{-1/2}V$ for a general parameter matrix V. As an approximate representation of the inverse operation the authors embed a number of steps of Newton's method into the network. Both methods, GroupSort and ONI, have the shortcoming that their orthogonalization schemes require the application of iterative computation schemes which incur a trade-off between the approximation quality and the computational cost.

For convolutional layers, more involved constructions are required to ensure that the resulting linear transformations are orthogonal. In particular, enforcing orthogonal kernel matrices is not sufficient in general to ensure a Lipschitz constant of 1 when the convolutions have overlapping receptive fields.

Skew Orthogonal Convolutions (SOC) [18] parameterize orthogonal matrices as the matrix exponentials of skew-symmetric matrices. They embed a truncation of the exponential's power series into the network and bound the resulting error. However, SOC requires a rather large number of iterations to yield good approximation quality, which leads to high computational cost.

Block Convolutional Orthogonal Parameterization (BCOP) [13] relies on a matrix decomposition approach to address the problem of orthogonalizing convolutional layers. The authors parameterize each convolution kernel of size $k \times k$ by a set of $2k-1$ convolutional matrices of size 1×2 or 2×1. These are combined with a final pointwise convolution with orthogonal kernel. However, BCOP also incurs high computation cost, because each of the smaller transforms requires orthogonalizing a corresponding parameter matrix.

Cayley Layers parameterize orthogonal matrices using the Cayley transform [5]. Naively, this requires the inversion of a matrix of size quadratic in the input dimensions. However, in [21] the author demonstrate that in certain situations, namely for full image size convolutions with circular padding, the computations can be performed more efficiently.

Explicitly Constructed Orthogonal Convolutions (ECO) [27] rely on a theorem that relates the singular values of the Jacobian of a circular convolution to the singular values of a set of much smaller matrices [16]. The authors derive a rather efficient parameterization that, however, is restricted to full-size dilated convolutions with non-overlapping receptive fields.

The main shortcomings of Cayley layers and ECO are their restriction to certain full-size convolutions. Those are incompatible with most well-performing network architectures for high-dimensional data, which use local kernel convolutions, such as 3×3, and overlapping receptive fields.

Relation to AOL. The AOL method that we detail in Sect. 4 can be seen as a hybrid of bound-based and orthogonality-based approaches. It mathematically guarantees the Lipschitz property of each network layer by rescaling the corresponding parameter matrix (column-wise for fully-connected layers, channel-wise for convolutions). In contrast to other bound-based approaches it does not use a computationally expensive iterative approach to estimate the Lipschitz constant as precisely as possible, but it relies on a closed-form upper bound. The bound is tight for matrices with orthogonal columns. During training this has the effect that orthogonal parameter matrices are implicitly preferred by the optimizer, because they allow fitting the data, and therefore minimizing the loss, the best. Consequently, AOL benefits from the advantages of orthogonality-based approaches, such as preserving the variance of the activations and the gradient magnitude, without the other methods' shortcomings of requiring difficult parameterizations and being restricted to specific layer types.

4 Almost-Orthogonal Lipschitz (AOL) Layers

In this section, we introduce our main contribution, almost-orthogonal Lipschitz (AOL) layers, which combine the advantages of rescaling and orthogonalization approaches. Specifically, we introduce a weight-dependent rescaling technique for the weights of a linear neural network layer that guarantees them to be 1-Lipschitz. It can be easily computed in closed form and is applicable to fully-connected as well as convolutional layers.

The main ingredient is the following theorem, which provides an elementary formula for controlling the spectral norm of a matrix by rescaling its columns.

Theorem 1. *For any matrix $P \in \mathbb{R}^{n\times m}$, define $D \in \mathbb{R}^{m\times m}$ as the diagonal matrix with $D_{ii} = \left(\sum_j \left|P^\top P\right|_{ij}\right)^{-1/2}$ if the expression in the brackets is non-zero, or $D_{ii} = 0$ otherwise. Then the spectral norm of PD is bounded by* 1.

Proof. The upper bound the spectral norm of PD follows from an elementary computation. By definition of the spectral norm, we have

$$\|PD\|^2_{\text{spec}} = \max_{\|\vec{v}\|_2=1} \|PD\vec{v}\|_2^2 = \max_{\|\vec{v}\|_2=1} \vec{v}^\top D^\top P^\top PD\vec{v}. \tag{3}$$

We observe that for any symmetric matrix $M \in \mathbb{R}^{n\times n}$ and any $w \in \mathbb{R}^n$:

$$\boldsymbol{w}^T M \boldsymbol{w} \leq \sum_{i,j=1}^n |M_{ij}||w_i||w_j| \leq \sum_{i,j=1}^n \frac{1}{2}|M_{ij}|(w_i^2 + w_j^2) = \sum_{i=1}^n \Big(\sum_{j=1}^n |M_{ij}|\Big) w_i^2 \tag{4}$$

where the second inequality follows from the general relation $2ab \leq a^2 + b^2$. Evaluating (4) for $M = P^\top P$ and $\boldsymbol{w} = D\vec{v}$, we obtain for all $\vec{v}$ with $\|\vec{v}\|_2 = 1$

$$\vec{v}^\top D^\top P^\top PD\,\vec{v} \leq \sum_{i=1}^n \Big(\sum_{j=1}^n |P^\top P|_{ij}\Big)(D_{ii}v_i)^2 \leq \sum_{i=1}^n v_i^2 = 1 \tag{5}$$

which proves the bound.

Note that when P has orthogonal columns of full rank, we have that $P^\top P$ is diagonal and $D = (P^\top P)^{-1/2}$, so $D^\top P^\top PD = I$, for I the identity matrix. Consequently, (5) holds with equality and the bound in Theorem 1 is tight.

In the rest of this section, we demonstrate how Theorem 1 allows us to control the Lipschitz constant of any linear layer in a neural network.

4.1 Fully-Connected Lipschitz Layers

We first discuss the case of fully-connected layers.

Lemma 1 (**Fully-Connected AOL Layers).** *Let $P \in \mathbb{R}^{n\times m}$ be an arbitrary parameter matrix. Then, the fully-connected network layer*

$$f(x) = Wx + b \tag{6}$$

is guaranteed to be 1-Lipschitz, when $W = PD$ for D defined as in Theorem 1.

Proof. The Lemma follows from Theorem 1, because the Lipschitz constant of f is bounded by the spectral norm of its Jacobian matrix, which is simply W.

Discussion. Despite its simplicity, there are a number aspects of Lemma 1 that are worth a closer look. First, we observe that a layer of the form $f(x) = PDx+b$

can be interpreted in two ways, depending on how we (mentally) put brackets into the linear term. In the form $f(x) = P(Dx)+b$, we apply an arbitrary weight matrix to a suitably rescaled input vector. In the form $f(x) = (PD)x + b$, we apply a column-rescaling operation to the weight matrix before applying it to the unchanged input. The two views highlight different aspects of AOL. The first view reflects the flexibility and high capacity of learning with an arbitrary parameter matrix, with only an intermediate rescaling operation to prevent the growth of the Lipschitz constant. The second view shows that AOL layers can be implemented without any overhead at prediction time, because the rescaling factors can be absorbed in the parameter matrix itself, even preserving potential structural properties such as sparsity patterns.

As a second insight from Lemma 1 we obtain how AOL relates to prior methods that rely on orthogonal weight matrices. As derived after Theorem 1, if the parameter matrix, P, has orthogonal columns of full rank, then $W = PD$ is an orthogonal weight matrix. In particular, when P is already an orthonormal matrix, then D will be the identity matrix, and W will be equal to P. Therefore, our method can express any linear map based on an orthonormal matrix, but it can also express other linear maps. If the columns of P are approximately orthogonal, in the sense that $P^\top P$ is approximately diagonal, then the entries of D are dominated by the diagonal entries of the product. The multiplication by D acts mostly as a normalization of the length of the columns of P, and the resulting W is an almost-orthogonal matrix.

Finally, observe that Lemma 1 does not put any specific numeric or structural constraints on the parameter matrix. Consequently, there are no restrictions on the optimizer or objective function when training AOL-networks.

4.2 Convolutional Lipschitz Layers

An analog of Lemma 1 for convolutional layers can, in principle, be obtained by applying the same construction as above: convolutions are linear operations, so we could compute their Jacobian matrix and determine an appropriate rescaling matrix from it. However, this naive approach would be inefficient, because it would require working with matrices that are of a size quadratic in the number of input dimensions and channels. Instead, by a more refined analysis, we obtain the following result.

Lemma 2 (Convolutional AOL Layers). *Let $P \in \mathbb{R}^{k\times k\times c_I\times c_O}$, be a convolution kernel matrix, where $k \times k$ is the* kernel size *and c_I and c_O are the number of input and output channels, respectively. Then, the convolutional layer*

$$f(x) = P * R(x) + b \tag{7}$$

is guaranteed to be 1-Lipschitz, where $R(x)$ is a channel-wise rescaling that multiplies each channel $c \in \{1, \dots, c_I\}$ of the input by

$$d_c = \Big(\sum_{i,j}\sum_{a=1}^{c_I}\Big|\sum_{b=1}^{c_O} P^{(a,b)} * P^{(c,b)}\Big|_{i,j}\Big)^{-1/2}. \tag{8}$$

We can equivalently write f as $f(x) = W * x + b$, where $W = P * D$ with $D \in \mathbb{R}^{1\times 1\times c_I \times c_I}$ given by $D_{1,1}^{(c,c)} = d_c$, and $D_{1,1}^{(c_1,c_2)} = 0$ for $c_1 \neq c_2$.

The proof consists of an explicit derivation of the Jacobian of the convolution operation as a linear map, followed by an application of Theorem 1. The main step is the explicit demonstration that the diagonal rescaling matrix can in fact be bounded by a per-channel multiplication with the result of a self-convolution of the convolution kernel. The details can be found in the supplemental material.

Discussion. We now discuss some favorable properties of Lemma 2. First, as in the fully-connected case, the rescaling operation again can be viewed either as acting on the inputs, or as acting on the parameter matrix. Therefore, the convolutional layer also combines the properties of high capacity and no overhead at prediction time. In fact, for 1 × 1 convolutions, the construction of Lemma 2 reduces to the fully-connected situation of Lemma 1.

Second, computing the scaling factors is efficient, because the necessary operations scale only with the size of the convolution kernel regardless of the image size. The rescaling preserves the structure of the convolution kernel, e.g. sparsity patterns. In particular, this means that constructs such as dilated convolutions are automatically covered by Lemma 2 as well, as these can be expressed as ordinary convolutions with specific zero entries.

Furthermore, Lemma 2 requires no strong assumption on the padding type, and works as long as the padding itself is 1-Lipschitz. Also, the computation of the scale factors is easy to implement in all common deep learning frameworks using batch-convolution operations with the input channel dimension taking the role of the batch dimension.

5 Experiments

We compare our method to related work in the context of *certified robust accuracy*, where the goal is to solve an image classification task in a way that provably prevents *adversarial examples* [20]. Specifically, we consider an input x as *certifiably robustly classified* by a model under input perturbations up to size ϵ, if $x+\delta$ is correctly classified for all δ with $\|\delta\| \leq \epsilon$. Then the *certified robust accuracy* of a classifier is the proportion of the test set that is certifiably robustly classified.

Consider a function f that generates a score for each class. Define the *margin* of f at input x with correct label y as

$$M_f(x) = \left[f(x)_y - \max_{i\neq y} f(x)_i\right]_+ \qquad \text{with} \qquad [\cdot]_+ = \max\{\cdot, 0\}. \tag{9}$$

Then the induced classifier, $C_f(x) = \operatorname{argmax}_i f(x)_i$, certifiably robustly classifies an input x if $M_f(x) > \sqrt{2}L\epsilon$, where L is the Lipschitz constant L of f. This relation can be used to efficiently determine (a lower bound to) the certified robust accuracy of Lipschitz networks [23].

Following prior work in the field, we conduct experiments that evaluate the certified robust accuracy for different thresholds, ϵ, on the CIFAR-10 as well

Table 2. Patchwise architecture. For all layers we use zero padding to keep the size the same. For AOL-Small we set w to 16, and we choose $w = 32$ and $w = 48$ for AOL-Medium and AOL-Large. Furthermore, l is the number of classes, and $l = 10$ for CIFAR-10 and $l = 100$ for CIFAR-100. *Concatenation Pooling* stacks all the inputs into a single vector, and *First channels* just selects the first channels and ignores the rest

Layer name	Kernel size	Stride	Activation	Output size	Amount
Concatenation Pooling	4×4	4×4	-	$8 \times 8 \times 48$	1
AOL Conv	1×1	1×1	MaxMin	$8 \times 8 \times 192$	1
AOL Conv	3×3	1×1	MaxMin	$8 \times 8 \times 192$	12
AOL Conv	1×1	1×1	None	$8 \times 8 \times 192$	1
First Channels	–	–	–	$8 \times 8 \times w$	1
Flatten	–	–	–	$64w$	1
AOL FC	–	–	MaxMin	$64w$	13
AOL FC	–	–	None	$64w$	1
First Channels	–	–	–	l	1

as the CIFAR-100 dataset. We also provide ablation studies that illustrate that AOL can be used in a variety of network architectures, and that it indeed learns matrices that are approximately orthogonal. In the following we describe our experimental setup. Further details can be found in the supplemental material. All hyperparameters were determined on validation sets.

Architecture: Our main model architecture is loosely inspired by the *ConvMixer* architecture [22]: we first subdivide the input image into 4×4 patches, which are processed by 14 convolutional layers, most of kernel size 3×3. This is followed by 14 fully connected layers. We report results for three different model sizes, we will refer to the models as AOL-Small, AOL-Medium and AOL-Large. The full architectural details can be found in Table 2.

Other network architectures are discussed in an ablation study in Sect. 6.1.

Initialization: In order to ensure stable training we initialize the parameter matrices so that our bound is tight. In particular, for layers preserving the size between input and output (e.g. the 3×3 convolutions) we initialize the parameter matrix so that the Jacobian is the identity matrix. For any other layers we initialize the parameter matrix so that it has random orthogonal columns.

Loss Function: In order to train the network to achieve good certified robust accuracy we want the score of the correct class to be bigger than any other score by a margin. We use a loss function similar to the one proposed for *Lipschitz-margin training* [23] with a temperature parameter that helps encouraging a margin during training. Our loss function takes as input the model's logit vector,

Table 3. Experimental results: robust image classification on CIFAR-10 for AOL and methods from the literature.Results for concurrent unpublished works (ECO and SOC with Householder activations) are printed in italics. *Standard CNN* refers to our implementation of a simple convolutional network trained without enforcing any robustness, for details see the supplemental material.

Method	Standard Accuracy	Certified Robust Accuracy			
		$\epsilon = \frac{36}{255}$	$\epsilon = \frac{72}{255}$	$\epsilon = \frac{108}{255}$	$\epsilon = 1$
Standard CNN	83.4%	0%	0%	0%	0%
BCOP Large [13]	72.2%	58.3%	–	–	–
GloRo 6C2F [12]	77.0%	58.4%	–	–	–
Cayley Large [21]	75.3%	59.2%	–	–	–
SOC-20 [18]	76.4%	61.9%	–	–	–
SOC-25 (from [27])	–	60.2%	43.7%	28.6%	–
ECO-25 [27]	*75.7 %*	*66.1 %*	*55.6 %*	*45.3 %*	–
SOC-15 (from [19]*)*	*76.4 %*	*63.0 %*	*48.5%*	*35.5 %*	–
AOL-Small	69.8%	62.0%	54.4%	47.1%	21.8%
AOL-Medium	71.1%	63.8%	56.1%	48.6%	23.2%
AOL-Large	71.6%	64.0%	56.4%	49.0%	23.7%

$\boldsymbol{s}$, as well as a one-hot encoding $\boldsymbol{y}$ of the true label as input, and is given by

$$\mathcal{L}(\boldsymbol{s}, \boldsymbol{y}) = \text{crossentropy}\Big(\boldsymbol{y},\ \text{softmax}\,\big(\frac{\boldsymbol{s} - u\boldsymbol{y}}{t}\big)t\Big), \tag{10}$$

for some offset u and some temperature t. For our experiments, we use $u = \sqrt{2}$, which encourages the model to learn to classify the training data certifiably robustly to perturbations of norm 1. Furthermore we use temperature $t = 1/4$, which causes the gradient magnitude to stay close to 1 as long as a training example is classified with margin less than 1/2.

Optimization: We minimize the loss function (10) using SGD with Nesterov momentum of 0.9 for 1000 epochs. The batch size is 250. The learning rate starts at 10^{-3} and is reduced by a factor of 10 at epochs $900, 990$ and 999. As data augmentation we use spatial transformations (rotations and flipping) as well as some color transformation. The details are provided in the supplemental material. For all AOL layers we also use weight decay with coefficient 5×10^{-4}.

6 Results

The main results can be found in Tables 3 and 4, where we compare the certified robust accuracy of our method to those reported in previous works on orthogonal networks and other networks with bounded Lipschitz constant. For methods that

Table 4. Experimental results: robust image classification on CIFAR-100 for AOL and methods from the literature.We report the standard accuracy on the test set as well as the certified robust accuracy under input perturbations up to size ϵ for different values of ϵ.Results for concurrent unpublished works (ECO and SOC with Householder activations) are printed in italics.

Method	Standard	Certified Robust Accuracy			
	Accuracy	$\epsilon = \frac{36}{255}$	$\epsilon = \frac{72}{255}$	$\epsilon = \frac{108}{255}$	$\epsilon = 1$
SOC-30 [18]	43.1%	29.2%	–	–	–
SOC (from [27])	–	28.6%	18.2%	10.9%	–
ECO-25 [27]	*41.7 %*	*32.6 %*	*25.1 %*	*19.2%*	–
SOC (from [19]*)*	*47.8 %*	*34.8 %*	*23.7 %*	*15.8 %*	–
AOL-Small	42.4%	32.5%	24.8%	19.2%	6.7%
AOL-Medium	43.2%	33.7%	26.0%	20.2%	7.2%
AOL-Large	43.7%	33.7%	26.3%	20.7%	7.8%

are presented in multiple variants, such as different networks depths, we include the variant for which the authors list results for large values of ϵ.

The table shows that our proposed methods achieves results comparable with the current state-of-the-art. For small robustness thresholds, it achieves certified robust accuracy on par with published earlier methods, though slightly below that reported in two concurrent preprints [19,27]. Focusing on (more realistic) medium or higher robustness thresholds, AOL achieves certified robust accuracy comparable to or even higher than all other methods. As a reference for future work, we also report values for an even higher robustness threshold than what appeared in the literature so far, $\epsilon = 1$.

Another observation is that on the CIFAR-10 dataset the clean accuracy of AOL is somewhat below other methods. We attribute this to the fact that we mainly focused our training towards high robustness. The accuracy-robustness trade-off can in fact be influenced by the choice of margin at training time, see our ablation study in Sect. 6.1.

6.1 Ablation Studies

In this section we report on a number of ablation studies that shed light on specific aspect of AOL.

Generality: One of the main advantages of AOL is that it is not restricted to a specific architecture or a specific layer type. To demonstrate this, we present additional experiments for a broad range of other architectures. AOL-FC consists simply of 9 fully connected layers. AOL-STD resembles a standard convolutional architecture, where the number of channels doubles whenever the resolution is reduced. AOL-ALT is another convolutional architecture that keeps the number of activations constant wherever possible in the network. AOL-DIL resembles

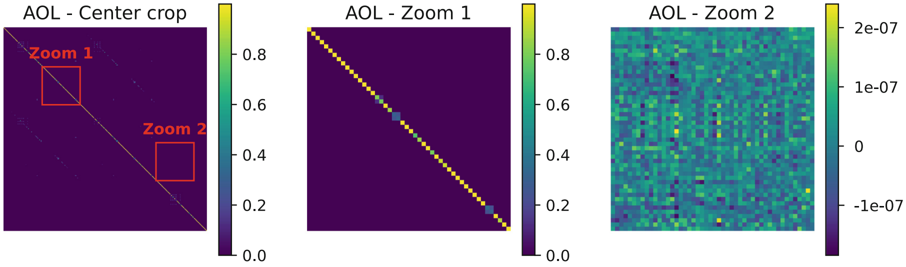

Fig. 1. Evaluation of the orthogonality of the trained model. We consider the third layer of the AOL-Small model. It is a 3×3 convolutions with input size $8 \times 8 \times 192$. We show a center crop of $J^\top J$, for J the Jacobian, as well as two further crops. Note that most diagonal elements are close to 1, and most off-diagonal elements are very close to 0 (note the different color scale in the third subplot). This confirms that AOL did indeed learn an almost-orthogonal weight matrix. Best viewed in color and zoomed in

the architectures used in [27] in that it uses large dilated convolutions instead of small local ones. It also uses circular padding. The details of the architectures are provided in the supplemental.

The results (shown in the supplemental material) confirm that for any of these architectures, we can train AOL-based Lipschitz networks and achieve certified robust accuracy comparable to the results of earlier specialized methods.

Approximate Orthogonality: As a second ablation study, we demonstrate that AOL indeed learns almost-orthogonal weight matrices, thereby justifying its name. In order to do that, we evaluate $J^\top J$ for J the Jacobian of an AOL convolution, and visualize it in Fig. 1. More detailed results including a comparison to standard training are provided in the supplemental material.

Table 5. Experimental results for AOL-Small for different value of u and t in the loss function in Equation (10).We report the standard accuracy on the test seWe report the standard accuracy on the test set as well as the certified robust accuracy under input perturbations up to size ϵ for different values of ϵ.t as well as the certified robust accuracy under input perturbations up to size ϵ for different values of ϵ.

u	t	Standard	Certified Robust Accuracy			
		Accuracy	$\epsilon = \frac{36}{255}$	$\epsilon = \frac{72}{255}$	$\epsilon = \frac{108}{255}$	$\epsilon = 1$
$\sqrt{2}/16$	1/64	79.8%	45.3%	16.7%	3.3%	0.0%
$\sqrt{2}/4$	1/16	77.4%	63.0%	47.6%	33.0%	2.5%
$\sqrt{2}$	1/4	70.4%	62.6%	55.0%	47.9%	22.2%
$4\sqrt{2}$	1	59.8%	55.5%	50.9%	46.5%	30.8%
$16\sqrt{2}$	4	48.2%	45.2%	42.2%	39.4%	28.6%

Accuracy-Robustness Tradeoff: The loss function in Eq. (10) allows trading off between clean accuracy and certified robust accuracy by changing the size of the enforced margin. We demonstrate this by an ablation study that varies the offset parameter u in the loss function, and also scales t proportional to u.

The results can be found in Table 5. One can see that using a small margin allows us to train an AOL Network with high clean accuracy, but decreases the certified robust accuracy for larger input perturbations, whereas choosing a higher offset allows us to reach state-of-the-art accuracy for larger input perturbations. Therefore, varying this offset gives us an easy way to prioritize the measure that is important for a specific problem.

7 Limitations

Despite its flexibility, AOL also has limitations. Some of those need to be overcome in order to enable training on high-resolution datasets in the future.

Firstly, while AOL in principle can handle skip connections, as they are used e.g. in ResNets, the bound will (generally) not be tight, and the network will lose dynamic range. We only recommend using skip connections for problems where that is acceptable.

Secondly, optimization seems to be harder for AOL layers. They take longer to converge than unconstrained layers, which manifests itself in more training epochs needed. We believe the reason for this is that the optimizer needs to keep the matrices approximately orthogonal in addition to fitting the training data, and that doing both takes more iterations. Also, local minima emerge, avoiding which needs a careful choice of initialization and learning rate.

Thirdly, AOL is designed for L^2-Lipschitzness. Consequenty, the robustness certificates also hold only for L^2-perturbations, and we can only give very weak guarantees for example for perturbations with bounded L^1 norm.

Finally, the rescaling factors are simple to compute using a matrix multiplication or a convolution. However, the complexity of this calculation grows with the number of channels of the input. A more detailed analysis of the computational complexity can be found in the supplemental material.

8 Conclusion

In this work, we proposed AOL, a method for constructing deep networks that have Lipschitz constant of at most 1 and therefore are robust against small changes in the input data. Our main contribution is a rescaling technique for network layers that ensures them to be 1-Lipschitz. It can be computed and trained efficiently, and is applicable to fully-connected and various types of convolutional layers. Training with the rescaled layers leads to weight matrices that are almost orthogonal without the need for a special parametrization and computationally costly orthogonalization schemes. We present experiments and ablation studies in the context of image classification with certified robustness. They show that AOL-networks achieve results comparable with methods that explicitly enforce orthogonalization, while offering the simplicity and flexibility of earlier bound-based approaches.

References

1. Anil, C., Lucas, J., Grosse, R.B.: Sorting out Lipschitz function approximation. In: International Conference on Machine Learning (ICML) (2019)
2. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: International Conference on Machine Learning (ICML) (2017)
3. Bank, D., Giryes, R.: An ETF view of dropout regularization. In: British Machine Vision Conference (BMVC) (2020)
4. Björck, Å., Bowie, C.: An iterative algorithm for computing the best estimate of an orthogonal matrix. SIAM J. Numer. Anal. **8**(2) (1971)
5. Cayley, A.: About the algebraic structure of the orthogonal group and the other classical groups in a field of characteristic zero or a prime characteristic. J. für die reine und angewandte Mathematik **30** (1846)
6. Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D.: Adversarial attacks and defences: a survey. arXiv preprint arXiv:1810.00069 (2018)
7. Cissé, M., Bojanowski, P., Grave, E., Dauphin, Y.N., Usunier, N.: Parseval networks: improving robustness to adversarial examples. In: International Conference on Machine Learning (ICML) (2017)
8. Cogswell, M., Ahmed, F., Girshick, R.B., Zitnick, L., Batra, D.: Reducing overfitting in deep networks by decorrelating representations. In: International Conference on Learning Representations (ICLR) (2016)
9. Gouk, H., Frank, E., Pfahringer, B., Cree, M.J.: Regularisation of neural networks by enforcing Lipschitz continuity. Mach. Learn. **110**(2), 393–416 (2020). https://doi.org/10.1007/s10994-020-05929-w
10. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of Wasserstein GANs. In: Conference on Neural Information Processing Systems (NeurIPS) (2017)
11. Huang, L., et al.: Controllable orthogonalization in training DNNs. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
12. Leino, K., Wang, Z., Fredrikson, M.: Globally-robust neural networks. In: International Conference on Machine Learning (ICML) (2021)
13. Li, B., Chen, C., Wang, W., Carin, L.: Certified adversarial robustness with additive noise. In: Conference on Neural Information Processing Systems (NeurIPS) (2019)
14. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: International Conference on Learning Representations (ICLR) (2018)
15. Saxe, A.M., McClelland, J.L., Ganguli, S.: Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In: International Conference on Learning Representations (ICLR) (2014)
16. Sedghi, H., Gupta, V., Long, P.M.: The singular values of convolutional layers. In: International Conference on Learning Representations (ICLR) (2019)
17. Serban, A., Poll, E., Visser, J.: Adversarial examples on object recognition: a comprehensive survey. ACM Comput. Surv. (CSUR) **53**(3), 1–38 (2020)
18. Singla, S., Feizi, S.: Skew orthogonal convolutions. In: International Conference on Machine Learning (ICML) (2021)

19. Singla, S., Feizi, S.: Improved deterministic l_2 robustness on CIFAR-10 and CIFAR-100. In: International Conference on Learning Representations (ICLR) (2022). https://openreview.net/forum?id=tD7eCtaSkR
20. Szegedy, C., et al.: Intriguing properties of neural networks. In: International Conference on Learning Representations (ICLR) (2014)
21. Trockman, A., Kolter, J.Z.: Orthogonalizing convolutional layers with the Cayley transform. In: International Conference on Learning Representations (ICLR) (2021)
22. Trockman, A., Kolter, J.Z.: Patches are all you need? arXiv preprint arXiv:2201.09792 (2022)
23. Tsuzuku, Y., Sato, I., Sugiyama, M.: Lipschitz-margin training: scalable certification of perturbation invariance for deep neural networks. In: Conference on Neural Information Processing Systems (NeurIPS) (2018)
24. Virmaux, A., Scaman, K.: Lipschitz regularity of deep neural networks: analysis and efficient estimation. In: Conference on Neural Information Processing Systems (NeurIPS) (2018)
25. Wang, J., Chen, Y., Chakraborty, R., Yu, S.X.: Orthogonal convolutional neural networks. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
26. Xu, H., et al.: Adversarial attacks and defenses in images, graphs and text: a review. Int. J. Autom. Comput. **17**(2), 151–178 (2020)
27. Yu, T., Li, J., CAI, Y., Li, P.: Constructing orthogonal convolutions in an explicit manner. In: International Conference on Learning Representations (ICLR) (2022). https://openreview.net/forum?id=Zr5W2LSRhD