Sorting Out Lipschitz Function Approximation

Cem Anil [*1 2] **James Lucas** [*1 2] **Roger Grosse** [1 2]

Abstract

Training neural networks under a strict Lipschitz constraint is useful for provable adversarial robustness, generalization bounds, interpretable gradients, and Wasserstein distance estimation. By the composition property of Lipschitz functions, it suffices to ensure that each individual affine transformation or nonlinear activation is 1-Lipschitz. The challenge is to do this while maintaining the expressive power. We identify a necessary property for such an architecture: each of the layers must preserve the gradient norm during backpropagation. Based on this, we propose to combine a gradient norm preserving activation function, GroupSort, with norm-constrained weight matrices. We show that norm-constrained GroupSort architectures are universal Lipschitz function approximators. Empirically, we show that norm-constrained GroupSort networks achieve tighter estimates of Wasserstein distance than their ReLU counterparts and can achieve provable adversarial robustness guarantees with little cost to accuracy.

Constraining the Lipschitz constant of a neural network puts a bound on how much its output can change in proportion to a change in its input. For classification tasks, a small Lipschitz constant leads to better generalization (Sokolić et al., 2017), improved adversarial robustness (Cisse et al., 2017; Tsuzuku et al., 2018), and greater interpretability (Tsipras et al., 2018). Additionally, the Wasserstein distance between two probability distributions can be expressed as the solution to a maximization problem over Lipschitz functions (Peyré & Cuturi, 2018). Despite the wide-ranging applications, the question of how to approximate Lipschitz functions with neural networks without sacrificing expressive power has remained largely unanswered.

[*]Equal contribution [1]Department of Computer Science, University of Toronto, Toronto, Canada [2]Vector Institute, Toronto, Canada. Correspondence to: Cem Anil <cem.anil@mail.utoronto.ca>, James Lucas <jlucas@cs.toronto.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

Existing approaches to enforce Lipschitz constraints fall into two categories: regularization and architectural constraints. Regularization approaches (Drucker & Le Cun, 1992; Gulrajani et al., 2017) perform well in practice, but do not provably enforce the Lipschitz constraint globally. Approaches based on architectural constraints place limitations on the operator norm (such as the matrix spectral norm) of each layer's weights (Cisse et al., 2017; Yoshida & Miyato, 2017). These provably satisfy the Lipschitz constraint, but come at a cost in expressive power. E.g., norm-constrained ReLU networks are provably unable to approximate simple functions such as absolute value (Huster et al., 2018).

We first identify a simple property that expressive norm-constrained Lipschitz architectures must satisfy: gradient norm preservation. Specifically, in order to represent a function with slope 1 almost everywhere, each layer must preserve the norm of the gradient during backpropagation. ReLU architectures satisfy this only when the activations are positive; empirically, this manifests during training of norm-constrained ReLU networks in that the activations are forced to be positive most of the time, reducing the network's capacity to represent nonlinear functions. We make use of an alternative activation function called *GroupSort* — a variant of which was proposed by Chernodub & Nowicki (2016) — which sorts groups of activations. GroupSort is both Lipschitz and gradient norm preserving. Using a variant of the Stone-Weierstrass theorem, we show that norm-constrained GroupSort networks are universal Lipschitz function approximators. While we focus our attention, both theoretically and empirically, on fully connected networks, the same general principles hold for convolutional networks where the techniques we introduce could be applied.

Empirically, we show that ReLU networks are unable to approximate even the simplest Lipschitz functions which GroupSort networks can. We observe that norm-constrained ReLU networks must trade non-linear processing for gradient norm, leading to less expressive networks. Moreover, we obtain tighter lower bounds on the Wasserstein distance between complex, high dimensional distributions using GroupSort architectures. We also train classifiers with provable adversarial robustness guarantees and find that using GroupSort provides improved accuracy and robustness compared to ReLU. Across all of our experiments, we found that norm-constrained GroupSort architectures consistently outperformed their ReLU counterparts.

2. Background

Notation We will use $\mathbf{x} \in \mathbb{R}^{in}$ to denote the input vector to the neural network, $\boldsymbol{y} \in \mathbb{R}^{out}$ the output (or logits), n_l the dimensionality of the l^{th} hidden layer, $\mathbf{W}_l \in \mathbb{R}^{n_{l-1} \times n_l}$ and $\boldsymbol{b}_l \in \mathbb{R}^{n_l}$ the weight matrix and the bias of the l^{th} layer. We will denote the pre-activations in layer l with $\boldsymbol{z}_l$ and activations with $\boldsymbol{h}_l$. The number of layers will be L with $\boldsymbol{y} = \boldsymbol{z}_L$. We will use ϕ to denote the activation used. The computation performed by layer l will be:

$$\boldsymbol{z}_l = \mathbf{W}_l \boldsymbol{h}_{l-1} + \boldsymbol{b}_l \qquad \boldsymbol{h}_l = \phi(\boldsymbol{z}_l)$$

2.1. Lipschitz Functions

Given two metric spaces $\mathcal{X}$ and $\mathcal{Y}$, a function $f : \mathcal{X} \to \mathcal{Y}$ is Lipschitz continuous if there exists $K \in \mathbb{R}$ such that for all x_1 and x_2 in $\mathcal{X}$,

$$d_{\mathcal{Y}}(f(x_1), f(x_2)) \leq K d_{\mathcal{X}}(x_1, x_2)$$

where $d_{\mathcal{X}}$ and $d_{\mathcal{Y}}$ are metrics (such as Euclidean distance) on $\mathcal{X}$ and $\mathcal{Y}$ respectively. In this work, when we refer to *the* Lipschitz constant we are referring to the smallest such K for which the above holds under a given $d_{\mathcal{X}}$ and $d_{\mathcal{Y}}$. Unless otherwise specified, we take $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Y} = \mathbb{R}^m$ throughout. If the Lipschitz constant of a function is K, it is called a *K-Lipschitz* function. If the function is everywhere differentiable then its Lipschitz constant is bounded by the operator norm of its Jacobian. Throughout this work, we make use of the following definition:

Definition 1. *Given a metric space (X, d_X) where d_X denotes the metric on X, we write $C_L(X, \mathbb{R})$ to denote the space of all 1-Lipschitz functions mapping X to $\mathbb{R}$ (with respect to the L_p metric).*

2.2. Lipschitz-Constrained Neural Networks

As 1-Lipschitz functions are closed under composition, to build a 1-Lipschitz neural network it suffices to compose 1-Lipschitz affine transformations and activations.

1-Lipschitz Linear Transformations: Ensuring that each linear map is 1-Lipschitz is equivalent to ensuring that $||\mathbf{W}\mathbf{x}||_p \leq ||\mathbf{x}||_p$ for any $\mathbf{x}$; this is equivalent to constraining the matrix p-norm, $||\mathbf{W}||_p = \sup_{||\mathbf{x}||_p=1} ||\mathbf{W}\mathbf{x}||_p$, to be at most 1. Important examples of matrix p-norms include the matrix 2-norm, which is the largest singular value, and the matrix ∞-norm, which can be expressed as:

$$||\mathbf{W}||_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^{m} |w_{ij}|.$$

Similarly, we may also define the mixed matrix norm, given by $||\mathbf{W}||_{p,q} = \sup_{||\mathbf{x}||_p=1} ||\mathbf{W}\mathbf{x}||_q$. Enforcing matrix norm constraints naively may be computationally expensive. We discuss techniques to efficiently ensure that $||W||_p = 1$ when $p = 2$ or $p = \infty$ in Section 4.2.

1-Lipschitz Activations: Most common activations (such as ReLU (Krizhevsky et al., 2012), tanh, maxout (Goodfellow et al., 2013)) are 1-Lipschitz, if scaled appropriately.

2.3. Applications of Lipschitz Networks

Wasserstein Distance Estimation Wasserstein-1 distance (also called Earth Mover Distance) is a way to compute the distance between two probability distributions and has found many applications in machine learning (Peyré & Cuturi, 2018). Using Kantorovich duality (Villani, 2008), one can recast the Wasserstein distance estimation problem as a maximization problem, defined over 1-Lipschitz functions:

$$W(P_1, P_2) = \sup_{f \in C_L(X, \mathbb{R})} \left(\mathop{\mathbb{E}}_{x \sim P_1}[f(x)] - \mathop{\mathbb{E}}_{x \sim P_2}[f(x)] \right) \quad (1)$$

Arjovsky et al. (2017) proposed the Wasserstein GAN architecture, which uses a Lipschitz network as its discriminator.

Adversarial Robustness Adversarial examples are inputs to a machine learning system which have been designed to force undesirable behaviour (Szegedy et al., 2013; Goodfellow et al., 2014). Given a classifier f and an input $\mathbf{x}$, we can write an adversarial example as $\mathbf{x}_{adv} = \mathbf{x} + \delta$ such that $f(\mathbf{x}_{adv}) \neq f(\mathbf{x})$ and δ is small. A small Lipschitz constant guarantees a lower bound on the size of δ (Tsuzuku et al., 2018), thus providing robustness guarantees.

Some Other applications Enforcing the Lipschitz constant on networks has found uses in regularization (Gouk et al., 2018) and stabilizing GAN training (Kodali et al., 2017).

3. Gradient Norm Preservation

When backpropagating through a norm-constrained 1-Lipschitz network, the gradient norm is non-increasing as it is processed by each layer. This leads to interesting consequences when we try to represent scalar-valued functions whose input-output gradient has norm 1 almost everywhere. (This relates to Wasserstein distance estimation, as an optimal dual solution has this property (Gulrajani et al., 2017).) To approximate such functions, the gradient norm must be preserved by each layer in the network during backpropagation. Unfortunately, norm-constrained networks with common activations are unable to achieve this.

Theorem 1. *Consider a neural net, $f : \mathbb{R}^n \to \mathbb{R}$, built with matrix 2-norm constrained weights ($||\mathbf{W}||_2 \leq 1$) and 1-Lipschitz, element-wise, monotonic activation functions. If $||\nabla f(\mathbf{x})||_2 = 1$ almost everywhere, then f is linear.*

A full proof is presented in Appendix D. As a special case, Theorem 1 shows that 2-norm-constrained networks with ReLU (or sigmoid, tanh, etc.) activations cannot represent the absolute value function. For ReLU layers, gradient norm can only be preserved if every activation is positive[1]. Hence, the network's input-output mapping must be linear.

[1]Except for units which don't affect the network's output.

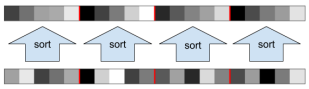

Figure 1: GroupSort activation with a grouping size of 5.

This tension between preserving gradient norm and nonlinear processing is also observed empirically. As the Lipschitz constant is decreased, the network is forced to sacrifice nonlinear processing capacity to maintain adequate gradient norm, as discussed later in Section 7.1.2 and Figure 6.

Another key observation is that we may adjust all weight matrices to have singular values of 1 without losing capacity [2].

Theorem 2. *Consider a network, $f : \mathbb{R}^n \to \mathbb{R}$, built with matrix 2-norm constrained weights and with $||\nabla f(\mathbf{x})||_2 = 1$ almost everywhere. Without changing the computed function, each weight matrix $\mathbf{W} \in R^{m\times k}$ can be replaced with a matrix $\tilde{\mathbf{W}}$ whose singular values all equal 1.*

The proof of Theorem 2 is given in Appendix D. The condition of singular values equaling 1 is equivalent to the following: when $m > k$, the columns of $\tilde{\mathbf{W}}$ are orthonormal; when $m < k$, the rows of $\tilde{\mathbf{W}}$ are orthonormal; and when $m = k$, $\tilde{\mathbf{W}}$ is orthogonal. We thereon refer to such matrices as orthonormal. With these in mind, we restrict our search for expressive Lipschitz networks to those that contain orthonormal weight matrices and activations which preserve the gradient norm during backpropagation.

4. Methods

If we can learn any 1-Lipschitz function with a neural network then we can trivially extend this to K-Lipschitz functions by scaling the output by K. We thus focus on designing 1-Lipschitz network architectures with respect to the L_2 and L_∞ metrics by requiring *each* layer to be 1-Lipschitz.

4.1. Gradient Norm Preserving Activation Functions

As discussed in Section 3, common activation functions such as ReLU are *not* gradient norm preserving. To achieve norm preservation, we use a general purpose 1-Lipschitz activation we call $\mathbf{GroupSort}$. This activation separates the pre-activations into groups, sorts each group into ascending order, and outputs the combined "group sorted" vector (shown in Figure 1). Visualizations of how GroupSort transforms pre-activations are shown in Appendix A.1.

Properties of GroupSort: GroupSort is a 1-Lipschitz operation. It is also norm preserving: its Jacobian is a permutation matrix, and permutation matrices preserve every vector p-norm. It is also homogeneous ($\mathbf{GroupSort}(\alpha\mathbf{x}) = \alpha\mathbf{GroupSort}(\mathbf{x})$) as sorting order is invariant to scaling.

[2] This condition implies gradient norm preservation.

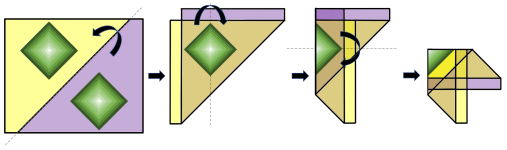

Figure 2: A rigid transformation, followed by absolute value, followed by another rigid transformation, can implement folding along an arbitrary hyperplane. Here, the network represents a function consisting of a pair of square pyramids by three folding operations, until the function is representable as a linear function of the top layer activations.

Varying the Grouping Size: When we pick a grouping size of 2, we call the operation **MaxMin**. This is equivalent to the Orthogonal Permutation Linear Unit (Chernodub & Nowicki, 2016). When sorting the *entire* input, we call the operation **FullSort**. MaxMin and FullSort are equally expressive: they can be reduced to each other without violating the norm constraint on the weights. FullSort can implement MaxMin by "chunking" the biases in pairs. We can write:

$$\mathbf{MaxMin}(\mathbf{x}) = \mathbf{FullSort}(\mathbf{I}\mathbf{x} + \boldsymbol{b}) - \boldsymbol{b},$$

where the biases $\boldsymbol{b}$ push each pair of activations to a different magnitude scale so that they get sorted independently (Appendix A.3). FullSort can also be represented using a series of MaxMin layers that implement BubbleSort; this obeys any matrix p-norm constraint since it can be implemented using only permutation matrices for weights. Although FullSort is able to represent certain functions more compactly, it is often more difficult to train compared to MaxMin.

Performing Folding via Absolute Value: Under the matrix 2-norm constraint, MaxMin is equivalent to absolute value in expressive power, as shown in Appendix A.3.

Applying absolute value to the activations has the effect of folding the space on each of the coordinate axes. Hence, a rigid linear transformation, followed by absolute value, followed by another rigid linear transformation, can implement folding along an arbitrary hyperplane. This gives an interesting interpretation of how MaxMin networks can represent certain functions by implementing absolute value, as shown in Figure 2. Montufar et al. (2014) provide an analysis of the expressivity of networks that can perform folding.

GroupSort and other activations: Without norm constraints, GroupSort can recover many other common activation functions. For example, ReLU, Leaky ReLU, concatenated ReLU (Shang et al., 2016), and Maxout (Goodfellow et al., 2013). Details can be found in Appendix A.2.

For a discussion regarding computational considerations, refer to Appendix A.4.

4.2. Norm-constrained linear maps

We discuss how to practically enforce the 1-Lipschitz constraint on the linear layers for both 2- and ∞-norms.

4.2.1. Enforcing $||W||_2 = 1$ while Preserving Gradient Norm

Several methods have been proposed to enforce matrix 2-norm constraints during training (Cisse et al., 2017; Yoshida & Miyato, 2017). In the interest of preserving the gradient norm, we go a step further and enforce *orthonormality* of the weight matrices. This stronger condition ensures that all singular values are exactly 1, rather than bounded by 1.

We make use of an algorithm first introduced by Björck & Bowie (1971), which we refer to as Björck Orthonormalization (or simply Björck). Given a matrix, this algorithm finds the closest orthonormal matrix through an iterative application of the Taylor expansion of the polar decomposition. Given an input matrix $A_0 = A$, the algorithm computes,

$$A_{k+1} = A_k \left(I + \frac{1}{2} Q_k + \ldots + (-1)^p \binom{-\frac{1}{2}}{p} Q_k^p \right) \quad (2)$$

where $Q_k = I - A_k^T A_k$. This algorithm is fully differentiable and thus has a pullback operator for the Stiefel manifold (Absil et al., 2009) allowing us to optimize over orthonormal matrices directly. A larger choice of p adds more computation but gives a closer approximation for each iteration. With $p = 1$ around 15 iterations is typically sufficient to give a close approximation but this is computationally prohibitive for wide layers. In practice, we found that we could use 2-3 iterations per forward pass and increase this to 15 or more iterations at the end of training to ensure a tightly enforced Lipschitz constraint. There exists a simple and easy-to-implement sufficient condition to ensure convergence, as described in Appendix B.3. Björck orthonormalization was also used by van den Berg et al. (2018) to enforce orthogonal weights for variational inference with normalizing flows (Rezende & Mohamed, 2015).

It is possible to project the weight matrices on the L2 ball both *after* each gradient descent step, or *during* forward pass (if the projection is differentiable). For the latter, the learnable parameters of the network are unconstrained during training. In our experiments, we exploit the differentiability of Björck operations and adopt the latter approach. Note that this doesn't lead to extra computational burden at test time, as the network parameters can be orthonormalized after training, and a new network can be built using these.

Other approaches have been proposed to enforce matrix 2-norm constraints. Parseval networks (Cisse et al., 2017) and spectral normalization (Miyato et al., 2018) are two such approaches, each of which can be used together with the GroupSort activation. Parseval networks also aim to set all of the singular values of the weight matrices to 1, and can be interpreted as a special case of Björck orthonormalization. In Appendix B.1 we provide a comparison of the Björck and Parseval algorithms. Spectral normalization is an inexpensive and practical way to enforce the 1-Lipschitz constraint, but since it only constrains the largest singular value to be less than 1, it is not gradient norm preserving by construction. We demonstrate the practical limitations caused by this with an experiment described in Appendix B.2.

While we restrict our focus to fully connected layers, our analyses apply to convolutions. Convolutions can be *unfolded* to be represented as linear transformations and bounding the spectral norm of the filters bound the spectral norm of the unfolded operation (Gouk et al., 2018; Cisse et al., 2017; Sedghi et al., 2018). For a discussion regarding computational considerations, refer to Appendix B.4.

4.2.2. Enforcing $||W||_\infty = 1$

Due to its simplicity and suitability for a GPU implementation, we use Algorithm 1 from Condat (2016) (see Appendix C) to project the weight matrices onto the L_∞ ball.

4.3. Provable Adversarial Robustness

A small Lipschitz constant limits the change in network output under small adversarial perturbations. As explored by Tsuzuku et al. (2018), we can guarantee adversarial robustness at a point by considering the *margin* about that point divided by the Lipschitz constant. Formally, given a network with Lipschitz constant K (with respect to the L_∞ metric) and an input $\mathbf{x}$ with corresponding class t that produces logits $\mathbf{y}$, we define its margin by

$$\mathcal{M}(\mathbf{x}) = \max(0, y_t - \max_{i \neq t} y_i) \quad (3)$$

If $\mathcal{M}(\mathbf{x}) > K\epsilon/2$, the network is robust to all perturbations δ with $||\delta||_\infty < \epsilon$, at $\mathbf{x}$. We train our networks with ∞-norm constrained weights using a multi-class hinge loss:

$$L(\mathbf{y}, t) = \sum_{i \neq t} \max(0, \kappa - (y_t - y_i)) \quad (4)$$

where κ controls the margin enforcement and depends on the Lipschitz constant and desired perturbation tolerance.

4.4. Dynamical Isometry and Preventing Vanishing Gradients

Gradient norm preserving networks can also represent functions whose input-output Jacobian has singular values that all concentrate near unity (Pennington et al., 2017), a property known as *dynamical isometry*. This property has been shown to speed up training by orders of magnitude when enforced during weight initialization (Pennington et al., 2017), and explored in the contexts of training RNNs (Chen et al., 2018) and deep CNNs (Xiao et al., 2018). Enforcing norm preservation on each layer also solves the vanishing gradients problem, as the norm of the back-propagated gradients are maintained at unity. Using our methods, one can achieve dynamical isometry throughout training (see Figure 16), reaping the aforementioned benefits. Note that ReLU networks cannot achieve dynamical isometry (Pennington et al., 2017). We leave exploring these benefits to a future study.

5. Related Work

Several methods have been proposed to train Lipschitz neural networks (Cisse et al., 2017; Yoshida & Miyato, 2017; Miyato et al., 2018; Gouk et al., 2018). Cisse et al. (2017) regularize the weights of the neural network to obey an orthonormality constraint. The corresponding update to the weights can be seen as one step of the Björck orthonormalization scheme (Eq. 2). This regularization can be thought of as projecting the weights closer to the manifold of orthonormal matrices after each update. This is a critical difference to our own work, in which a differentiable projection is used during each update. Another approach, spectral normalization (Miyato et al., 2018), employs power iteration to rescale each weight by its spectral norm. Although efficient, spectral normalization doesn't guarantee gradient norm preservation and can therefore under-use Lipschitz capacity, as discussed in Appendix B.2. Arjovsky et al. (2016); Wisdom et al. (2016); Sun et al. (2017) use explicitly parametrized square orthogonal weight matrices.

Other techniques penalize the Jacobian of the network, constraining the Lipschitz constant locally (Gulrajani et al., 2017; Drucker & Le Cun, 1992; Sokolić et al., 2017). While it is often easy to train networks under such penalties, these methods don't provably enforce a Lipschitz constraint.

The Lipschitz constant of neural networks has been connected theoretically and empirically to generalization performance (Bartlett, 1998; Bartlett et al., 2017; Neyshabur et al., 2017; 2018; Sokolić et al., 2017). Neyshabur et al. (2018) show that if the network Lipschitz constant is small then a non-vacuous bound on the generalization error can be derived. Small Lipschitz constants have also been linked to adversarial robustness (Tsuzuku et al., 2018; Cisse et al., 2017). In fact, adversarial training can be viewed as approximate gradient regularization (Miyato et al., 2017; Simon-Gabriel et al., 2018) which makes the function Lipschitz locally around the training data. Lipschitz constants can also be used to provide provable adversarial robustness guarantees. Tsuzuku et al. (2018) manually enforce a margin depending on an approximation of the upper bound on the Lipschitz constant which in turn guarantees adversarial robustness. In this work we also explore provable adversarial robustness through margin training but do so with a network whose Lipschitz constant is known and globally enforced.

Classic neural network universality results use constructions which violate the norm-constraints needed for Lipschitz guarantees (Cybenko, 1989; Hornik, 1991). Huster et al. (2018) explored universal approximation properties of Lipschitz networks and proved that ReLU networks cannot approximate absolute value with ∞-norm constraints. In this work we also show that many activations, including ReLU, are not sufficient with 2-norm constraints. We prove that Lipschitz functions *can* be universally approximated if the correct activation function is used.

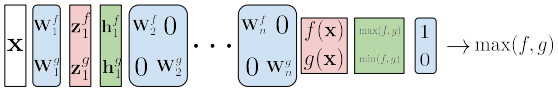

Figure 3: Lattice construction for universal approximation.

6. Universal Approximation of Lipschitz Functions

Universal approximation results for continuous functions don't apply to Lipschitz networks as the constructions typically involve large Lipschitz constants. Moreover, Huster et al. (2018) showed that it is impossible to approximate even the absolute value function with ∞-norm-constrained ReLU networks. We now present theoretical guarantees on the approximation of Lipschitz functions. To our knowledge, this is the first universal Lipschitz function approximation result for norm-constrained networks.

We will first prove a variant of the Stone-Weierstrass Theorem which gives a simple criterion for universality (similar to Lemma 4.1 in Yaacov (2010)). We then construct a class of networks with GroupSort which satisfy this criterion.

Definition 2. *We say that a set of functions, L, is a* lattice *if for any $f, g \in L$ we have $max(f, g) \in L$ and $min(f, g) \in L$ (where max and min are defined pointwise).*

Lemma 1. *(Restricted Stone-Weierstrass Theorem) Suppose that (X, d_X) is a compact metric space with at least two points and L is a lattice in $C_L(X, \mathbb{R})$ with the property that for any two distinct elements $x, y \in X$ and any two real numbers a and b such that $|a - b| \leq d_X(x, y)$ there exists a function $f \in L$ such that $f(x) = a$ and $f(y) = b$. Then L is dense in $C_L(X, \mathbb{R})$.*

Remark. *We could replace $|\cdot|$ with any metric on $\mathbb{R}$.*

The full proof of Lemma 1 is presented in Appendix E. Note that Lemma 1 says that $\mathcal{A}$ is a universal approximator for 1-Lipschitz functions iff $\mathcal{A}$ is a lattice that separates points. Using Lemma 1, we can derive the second of our key results. Norm-constrained networks with GroupSort activations are able to approximate any Lipschitz function in L_p distance.

Theorem 3. *(Universal Approximation with Lipschitz Networks) Let $\mathcal{LN}_p$ denote the class of fully-connected networks whose first weight matrix satisfies $||\mathbf{W}_1||_{p,\infty} = 1$, all other weight matrices satisfy $||\mathbf{W}||_\infty = 1$, and GroupSort activations have a group size of 2. Let X be a closed and bounded subset of $\mathbb{R}^n$ endowed with the L_p metric. Then the closure of $\mathcal{LN}_p$ is dense in $C_L(X, \mathbb{R})$.*

Proof. (Sketch) Observe first that $\mathcal{LN}_p \subset C_L(X, \mathbb{R})$. By Lemma 1, it is sufficient to show that $\mathcal{LN}_p$ is closed under max and min and has the point separation property. For the latter, given $x, y \in X$ and $a, b \in \mathbb{R}$ with $|a-b| \leq ||x-y||_p$, we can fit a line with a single layer network, f, satisfying the 1-Lipschitz constraint with $f(x) = a$ and $f(x) = b$.

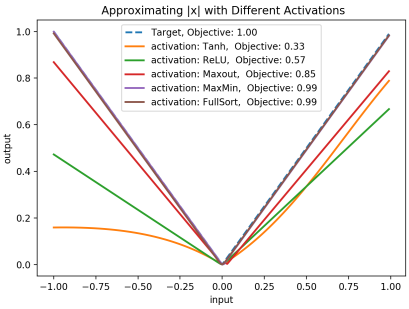

Figure 4: Approximating the absolute value function with Lipschitz networks using different activations. The objective values indicate the estimated Wasserstein Distance.

Now consider f and g in $\mathcal{LN}_p$. For simplicity, assume that they have the same number of layers. We can construct the layers of another network $h \in \mathcal{LN}_p$ by vertically concatenating the weight matrices of the first layer in f and g, followed with block diagonal matrices constructed from the remaining layers of f and g (see Figure 3). The final layer of the network computes $[f(x), g(x)]$. We then apply GroupSort to get $[max(f,g)(x), min(f,g)(x)]$ and take the dot product with $[1,0]$ or $[0,1]$ to get the max or min. □

The formal proof of Theorem 3 is presented in Appendix E. One special case of Theorem 3 is for 1-Lipschitz functions in L_∞ norm, in this case we may extend the restricted Stone-Weierstrass theorem in L_∞ norm to vector-valued functions to prove universality in this setting. Formally:

Observation. *Consider the set of networks, $\mathcal{LN}_\infty^m = \{f : \mathbb{R}^n \to \mathbb{R}^m, ||W||_\infty = 1\}$. Then $\mathcal{LN}_\infty^m$ is dense in 1-Lipschitz functions with respect to the L_∞ metric.*

While these constructions rely on constraining the ∞-norm of the weights [3], constraining the 2-norm often makes the networks easier to train, and we have not yet found a Lipschitz function which 2-norm constrained GroupSort networks couldn't approximate empirically. It remains an open question whether 2-norm constrained GroupSort networks are also universal Lipschitz function approximators.

7. Experiments

Our experiments had two main goals. First, we wanted to test whether norm-constrained GroupSort architectures can represent Lipschitz functions other approaches cannot. Second, we wanted to test if our networks can perform competitively with existing approaches on practical tasks that require strict bounds on the global Lipschitz constant. We present additional results in Appendix G, including CIFAR-10 (Krizhevsky, 2009) classification and MNIST small data classification. Experiment details are shown in Appendix H.

[3] Our construction fails for 2-norm constrained weights, as column-wise stacking two matrices that have max singular values of 1 might result in a matrix that has singular values larger than 1.

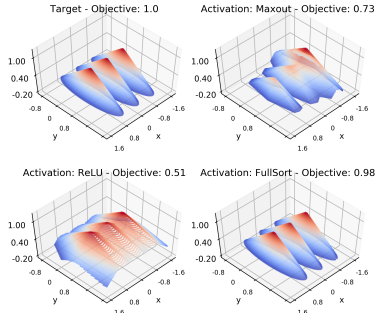

Figure 5: Approximating three circular cones with slope 1 using Lipschitz networks using various activations. The objective values represent the estimated Wasserstein distance.

7.1. Representational Capacity

We investigate the ability of 2-norm-constrained networks with different activations to represent Lipschitz functions.

7.1.1. Quantifying Expressive Power

We propose an effective method to quantify how expressive different Lipschitz architectures are. We pick pairs of probability distributions whose Wasserstein Distance and optimal dual surfaces can be computed analytically. We then train networks under the Wasserstein distance objective (Equation 1) using samples from these distributions to assess how closely they can estimate the Wasserstein distance. For 1D and 2D problems, we visualize the learned dual surfaces to inspect the failure modes of non-expressive architectures.

We focus on approximating the absolute value function, three circular cones and circular cones in higher dimensions. Appendix H.1 describes how pairs of probability distributions can be picked which have these optimal dual surfaces, and a Wasserstein distance of precisely 1.

Figure 4 shows the functions learned by Lipschitz networks built with various activations, trained to approximate absolute value. Non-GNP (non-gradient norm preserving) activations are incapable of approximating this trivial Lipschitz function. While increasing the network depth helps (Table 1), this representational barrier leads to limitations as the problem dimensionality increases.

Figure 5 shows the dual surfaces approximated by networks trained to approximate three circular cones. This figure points to an even more serious pathology with non-GNP activations: by attempting to increase the slope, the non-GNP networks may distort the shape of the function, leading to different behavior from the optimal solution. In the case of training WGAN critics, this cannot be fixed by increasing the Lipschitz constant. (Optimal critics for different Lipschitz constants are equivalent up to scaling.)

We evaluated the expressivity of architectures built with different activations for higher dimensional inputs, on the task of approximating high dimensional circular cones. As

shown in Table 1, increasing problem dimensionality leads to significant drops in the Wasserstein objective for networks built with non-GNP activations, and increasing the depth of the networks only slightly improves the situation. We also observed that while the MaxMin activation performs significantly better, it also needs large depth to learn the optimal solution. Surprisingly, shallow FullSort networks can easily approximate high dimensional circular cones.

7.1.2. Relevance of Gradient Norm Preservation in Practical Settings

Thus far, we have focused on examples where the gradient of the network should be 1 almost everywhere. For many practical tasks we don't need to meet this strong condition. Is gradient norm preservation relevant in other settings?

How much of the Lipschitz capacity can we use? We have proven that ReLU networks approach linear functions as they utilize the full gradient capacity allowed with Lipschitz constraints. To understand these implications practically, we trained ReLU and GroupSort networks on MNIST with orthonormal weight constraints enforced to ensure that they are 10-Lipschitz functions. We looked at the distribution of the spectral radius (largest singular value) of the network Jacobian over the training data. Figure 6 displays this distribution for each network. We observed that while both networks satisfy the Lipschitz constraint, the GroupSort network does so much more tightly than the ReLU network. The ReLU network was not able to make use of the capacity afforded to it and the observed Lipschitz constant was actually closer to 8 than 10. In Appendix G.3 we show the full singular value distribution which suggests that 2-norm-constrained GroupSort networks can achieve near-dynamical isometry throughout training.

We studied the activation statistics of ReLU networks trained on MNIST with and without Lipschitz constraints in Figure 6. Given a threshold, $\tau \in [0, 1]$, we computed the proportion of activations throughout the network which are positive at least as often as τ over the training data. Without a Lipschitz constraint, the activation statistics were much sparser, with almost no units active when $\tau > 0.4$. Smaller

Input Dim.	128	128	256	256	512	512
Depth	*3*	*7*	*3*	*7*	*3*	*7*
ReLU	0.51	0.60	0.50	0.53	0.46	0.49
Maxout	0.66	0.71	0.60	0.66	0.52	0.56
MaxMin	0.87	0.95	0.83	0.93	0.72	0.88
FullSort	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

Table 1: **Effect of problem dimensionality:** Testing how well different activations and depths can optimize the dual Wasserstein objective with different input dimensionality. The optimal dual surface obtains a dual objective of 1.

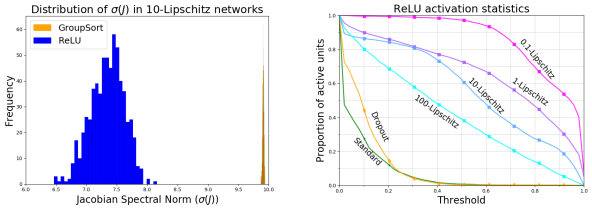

Figure 6: **(left) Jacobian spectral norm distribution:** Jacobian spectral norm distribution of 10 Lipschitz ReLU and GroupSort nets. **(right) Activation statistics Lipschitz ReLU nets:** Ratio of activations that are positive more often than the threshold.

Lipschitz constants forced the network to use more positive activations to make use of its gradient capacity (see Section 3). In the worst case, about 10% of units were "undead", or active all of the time, and hence didn't contribute any nonlinear processing. It's not clear what effect this has on representational capacity, but such a dramatic change in the network's activation statistics suggests that it made significant compromises to maintain adequate gradient norm.

7.2. Wasserstein Distance Estimation

We have shown that our methods can obtain tighter lower bounds on Wasserstein distance on synthetic tasks in Section 7.1.1. We now consider the more challenging task of computing the Wasserstein distance between the generator distribution of a GAN and the empirical distribution of the data it was trained on.[4] As the optimal surfaces under the dual Wasserstein objective have a gradient norm of 1 almost everywhere (Corollary 1 in Gemici et al. (2018)), the gradient norm preservation properties discussed in Section 3 are critical. Experiment details are outlined in Appendix H.2.

[4]The Wasserstein distance to the empirical data distribution is likely to be a loose upper bound on the Wasserstein distance to the data generating distribution, but this task still tests the ability to estimate Wasserstein distance in high-dimensional spaces.

	Linear	**MNIST**	**CIFAR10**
ReLU	Spectral	0.95 ± 0.01	1.12 ± 0.02
Maxout	Spectral	1.20 ± 0.03	1.40 ± 0.01
MaxMin	Spectral	1.36 ± 0.07	1.62 ± 0.04
GroupSort(4)	Spectral	1.64 ± 0.02	1.63 ± 0.03
GroupSort(9)	Spectral	1.70 ± 0.02	1.41 ± 0.04
ReLU	Björck	1.40 ± 0.01	1.39 ± 0.01
Maxout	Björck	1.95 ± 0.01	1.76 ± 0.02
MaxMin	Björck	2.16 ± 0.01	2.08 ± 0.02
GroupSort(4)	Björck	$\mathbf{2.31 \pm 0.01}$	2.17 ± 0.02
GroupSort(9)	Björck	2.31 ± 0.01	$\mathbf{2.23 \pm 0.02}$

Table 2: Estimating the Wasserstein Distance between the data and generator distributions using 1-Lipschitz feedforward networks, for MNIST and CIFAR10 GANs.

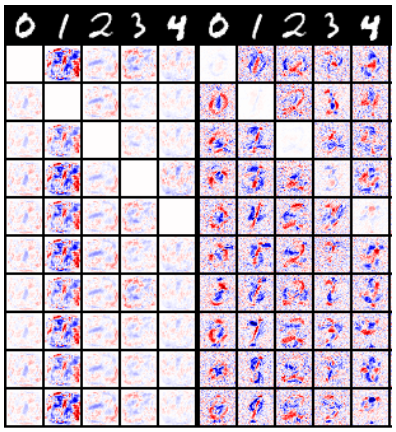

Figure 7: Gradients of input images with respect to targeted cross-entropy loss, for standard (left) and Lipschitz (right) nets. Images from classes 0-4 were chosen randomly from the test set (first row). Following rows show the gradient with different targets (0-9). Positive pixel values are red.

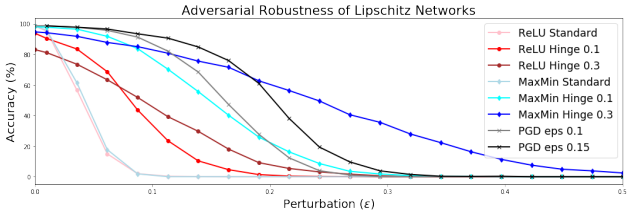

Figure 8: **Adversarial Robustness** Accuracy on PGD adversarial examples for varying perturbation sizes ϵ.

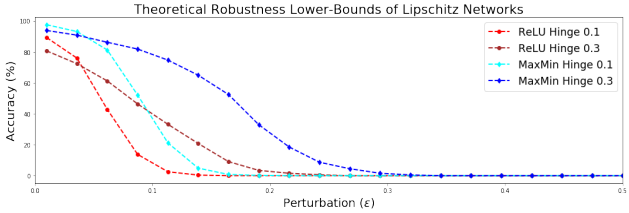

Figure 9: **Theoretical Adversarial Robustness** Theoretical accuracy lower bound for perturbation sizes ϵ.

We trained a GAN variant on MNIST and CIFAR10 datasets, then froze the weights of the generators. Using samples from the generator and original data distribution, we trained independent 1-Lipschitz networks to compute the Wasserstein distance between the empirical data distribution and the generator distribution. We used a shallow fully connected architecture (3 layers, 720 neurons wide). As seen in Table 2, using norm-preserving activation functions helps achieve tighter lower bounds on Wasserstein distance.

Training WGANs: We were also able to train Wasserstein GANs (Arjovsky et al., 2017) whose discriminators comprised of networks built with our proposed proposed 1-Lipschitz building blocks. Some generated samples can be found in Appendix H.3. We leave further investigation of the GANs built with our techniques to a future study.

7.3. Robustness and Interpretability

We explored the robustness of Lipschitz networks trained on MNIST to adversarial perturbations measured with L_∞ distance. We enforced an L_∞ constraint on the weights and used the multi-class hinge loss (Equation 4), as we found this to be more effective than manual margin training (Tsuzuku et al., 2018). We enforced a Lipschitz constant of $K = 1000$ and chose the margin $\kappa = Ka$ where a was 0.1 or 0.3. This technique provides margin-based provable robustness guarantees as described in Section 4.3. We also compared to PGD training (Madry et al., 2017). We attacked each model using the FGS and PGD methods (using random restarts and 200 iterations for the latter) (Szegedy et al., 2013; Madry et al., 2017) under the CW loss (Carlini & Wagner, 2016). The results are presented in Figure 8. The Lipschitz networks with MaxMin activations achieved better clean accuracy and larger margins than their ReLU counterparts, leading to better robustness. PGD training requires large capacity networks Madry et al. (2017) and we were unable to match the large perturbation performance of margin training with this architecture (using a larger CNN would produce better results). Note that the Lipschitz networks don't see any adversarial examples during training.

With the strictly enforced Lipschitz constant, we can compute theoretical lower bounds on the accuracy against adversaries with a maximum perturbation strength ϵ. In Figure 9, we show this lower bound for each of the models previously studied. This is computed by finding the proportion of data points which violate the margin by at least $K\epsilon$. Note that at the computed threshold, the model has low confidence in the adversarial example. An even larger perturbation would be required to induce confident misclassification.

Adversarially trained networks learn robust features and have interpretable gradients (Tsipras et al., 2018). We found that this holds for Lipschitz networks, without using adversarial training. The gradients with respect to the inputs are displayed for a standard network and a Lipschitz network (with 2-norm constraints) in Figure 7.

8. Conclusion

We identified gradient norm preservation as a critical component of Lipschitz network design and showed that failure to achieve this leads to less expressive networks. By combining the GroupSort activation and orthonormal weight matrices, we presented a class of networks which are provably 1-Lipschitz and can approximate any 1-Lipschitz function arbitrarily well. Empirically, we showed that our GroupSort networks are more expressive than existing architectures and can be used to achieve better estimates of Wasserstein distance and provable adversarial robustness guarantees.

ACKNOWLEDGMENTS

We extend our warm thanks to our colleagues for many helpful discussions. In particular, we would like to thank Mufan Li for pointing us towards the lattice formulation of the Stone-Weierstrass theorem, and Qiyang Li for his help in correcting a minor issue with the robustness experiments. We also thank Elliot Creager, Ethan Fetaya, Jörn Jacobsen, Mark Brophy, Maryham Mehri Dehnavi, Philippe Casgrain, Saeed Soori, Xuchan Bao and many others not listed here for draft feedback and many helpful conversations.

References

Absil, P.-A., Mahony, R., and Sepulchre, R. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

Arjovsky, M., Shah, A., and Bengio, Y. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pp. 1120–1128, 2016.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

Bartlett, P. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pp. 6240–6249, 2017.

Björck, Å. and Bowie, C. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM Journal on Numerical Analysis*, 8(2):358–364, 1971.

Bruna, J. and Mallat, S. Invariant scattering convolution networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35 (8):1872–1886, August 2013. ISSN 0162-8828. doi: 10.1109/TPAMI.2012.230. URL http://dx.doi.org/10.1109/TPAMI.2012.230.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. *arXiv preprint arXiv:1608.04644*, 2016.

Chen, M., Pennington, J., and Schoenholz, S. S. Dynamical isometry and a mean field theory of rnns: Gating enables signal propagation in recurrent neural networks. *arXiv preprint arXiv:1806.05394*, 2018.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pp. 2172–2180, 2016.

Chernodub, A. and Nowicki, D. Norm-preserving orthogonal permutation linear unit activation functions (oplu). *arXiv preprint arXiv:1604.02313*, 2016.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. Parseval networks: Improving robustness to adversarial examples. *arXiv preprint arXiv:1704.08847*, 2017.

Condat, L. Fast projection onto the simplex and the l_1 ball. *Mathematical Programming*, 158(1-2):575–585, 2016.

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

Drucker, H. and Le Cun, Y. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6):991–997, 1992.

Gemici, M., Akata, Z., and Welling, M. Primal-dual Wasserstein GAN. *arXiv preprint arXiv:1805.09575*, 2018.

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. Maxout networks. In Dasgupta, S. and McAllester, D. (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1319–1327, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL http://proceedings.mlr.press/v28/goodfellow13.html.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014.

Gouk, H., Frank, E., Pfahringer, B., and Cree, M. Regularisation of neural networks by enforcing lipschitz continuity. *arXiv preprint arXiv:1804.04368*, 2018.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

Hasenclever, L., Tomczak, J. M., van den Berg, R., and Welling, M. Variational inference with orthogonal normalizing flows. 2017.

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

Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Huster, T., Chiang, C.-Y. J., and Chadha, R. Limitations of the Lipschitz constant as a defense against adversarial examples. *arXiv preprint arXiv:1807.09705*, 2018.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kodali, N., Abernethy, J., Hays, J., and Kira, Z. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*, 2017.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

Lucas, J., Zemel, R., and Grosse, R. Aggregated momentum: Stability through passive damping. *arXiv preprint arXiv:1804.00325*, 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pp. 2924–2932, 2014.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5947–5956, 2017.

Neyshabur, B., Bhojanapalli, S., and Srebro, N. A PAC-bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Skz_WfbCZ.

Nvidia, C. Programming guide, 2010.

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

Pennington, J., Schoenholz, S., and Ganguli, S. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in neural information processing systems*, pp. 4785–4795, 2017.

Peyré, G. and Cuturi, M. Computational optimal transport. *arXiv preprint arXiv:1803.00567*, 2018.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning-Volume 37*, pp. 1530–1538. JMLR. org, 2015.

Sedghi, H., Gupta, V., and Long, P. M. The singular values of convolutional layers. *arXiv preprint arXiv:1805.10408*, 2018.

Shang, W., Sohn, K., Almeida, D., and Lee, H. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *International Conference on Machine Learning*, pp. 2217–2225, 2016.

Simon-Gabriel, C.-J., Ollivier, Y., Schölkopf, B., Bottou, L., and Lopez-Paz, D. Adversarial vulnerability of neural networks increases with input dimension. *arXiv preprint arXiv:1802.01421*, 2018.

Sokolić, J., Giryes, R., Sapiro, G., and Rodrigues, M. R. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017.

Sun, S., Chen, C., and Carin, L. Learning structured weight uncertainty in bayesian neural networks. In *Artificial Intelligence and Statistics*, pp. 1283–1292, 2017.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

Townsend, J. A new trick for calculating Jacobian vector products. https://j-towns.github.io/2017/06/12/A-new-trick.html, 2017.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. There is no free lunch in adversarial robustness (but there are unexpected benefits). *arXiv preprint arXiv:1805.12152*, 2018.

Tsuzuku, Y., Sato, I., and Sugiyama, M. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. *arXiv preprint arXiv:1802.04034*, 2018.

van den Berg, R., Hasenclever, L., Tomczak, J. M., and Welling, M. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

Villani, C. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Wisdom, S., Powers, T., Hershey, J., Le Roux, J., and Atlas, L. Full-capacity unitary recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 4880–4888, 2016.

Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S. S., and Pennington, J. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. *arXiv preprint arXiv:1806.05393*, 2018.

Yaacov, I. B. Lipschitz functions on topometric spaces. *arXiv preprint arXiv:1010.1600*, 2010.

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

Zagoruyko, S. and Komodakis, N. Wide residual networks. *CoRR*, abs/1605.07146, 2016.