UNIVERSAL LIPSCHITZ NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Training neural networks with a Lipschitz constraint provides improved generalization, robustness, and interpretability. However, existing techniques either fail to guarantee a Lipschitz constraint or are unable to universally approximate Lipschitz functions. Often, a small Lipschitz constant is enforced by considering constraints on the network weights, but little attention is payed to the choice of activation function. We identify *Jacobian norm* of network layers as a scarce resource in representing Lipschitz functions and show that common activation functions are unable to effectively utilize this. We show that with common activation functions networks are unable to learn even the simplest Lipschitz functions, such as the absolute value function. With this insight, we introduce a novel activation function, the $GroupSort$ activation, which partitions the hidden layer and sorts the units within each partition. Empirically, we identify pathologies of common activation functions and confirm that these theoretical observations are relevant in practice.

Recently, there has been a surge of interest in training neural networks which have a Lipschitz property. The Lipschitz property ensures that a small change in the network input can only produce a small change in its output, as decided by a Lipschitz constant. For classification, one can show that a small Lipschitz constant leads to better generalization, improved adversarial robustness, and greater interpretability (Sokolić et al., 2017; Tsipras et al., 2018). The need for such networks extends beyond classification. Computing the Wasserstein distance — a principled way of computing the distance between two probability distributions — can be rewritten in its dual form as a search over Lipschitz functions. The WGAN (Arjovsky et al., 2017) uses this formulation in its training objective and the question of how to optimize over Lipschitz functions has played a major role in the development of these techniques (Gulrajani et al., 2017; Gemici et al., 2018).

Despite this interest, the question of how to restrict neural networks to be Lipschitz functions remains largely unanswered. A common approach adds a gradient penalty to the loss function (Drucker & Le Cun, 1992; Gulrajani et al., 2017). Although this method seems to work well in practice it does not provide guarantees on the Lipschitz constant. Other approaches, so called *atomic* constructions, directly restrict the architecture by placing limitations on the L_p norm (such as the matrix spectral norm) of each layer's weight matrix (Cisse et al., 2017; Yoshida & Miyato, 2017). Outside of practical limitations, these techniques can guarantee a small Lipschitz constant but they raise a new question: how restrictive are these weight constraints? Unfortunately, it turns out that these networks are typically unable to universally approximate Lipschitz functions (Huster et al., 2018).

In this work, we provide concrete answers to the question of universal approximation of Lipschitz functions. We further study the issues raised in Huster et al. (2018) and expand their negative results for additional norms and activation functions. In doing so, we identify that for an atomic construction to have the expressiveness to represent a large class of Lipschitz function, it must use Jacobian norm preserving layers. That is, the Jacobian of each linear transformation and activation should not allow the reduction of Jacobian of the entire neural network.

Our primary contributions are as follows. We identify Jacobian norm preservation as an important design principle in building universal Lipschitz function approximators and introduce the novel *GroupSort* activation as a general means to achieve this. We demonstrate that using common activations such as ReLU within Lipschitz constrained networks leads to pathologies in the learned networks but show that these issues are resolved by GroupSort. Using the Stone-Weierstrass theorem, we present simple criteria by which one can build Lipschitz neural networks which are universal function approximators. We explicitly construct one such class of networks. Finally, we

explore applications of Lipschitz neural networks to Wasserstein distance estimation, classification, and robustness.

2 BACKGROUND

Notation We will use $\boldsymbol{x} \in \mathbb{R}^{in}$ as the input vector to the neural network, $n^{(l)}$ as the dimensionality of the lth hidden layer, $\mathbf{W^{(l)}} \in \mathbb{R}^{\mathbf{n^{(l-1)}} \times \mathbf{n^{(l)}}}$ and $\boldsymbol{b}^{(l)} \in \mathbb{R}^{n^{(l)}}$ as the weight matrix and the bias of the lth layer. The number of layers in the network will be L. We will denote the pre-activations in each layer with $\boldsymbol{z}^{(l)}$ and activations with $\boldsymbol{h}^{(l)}$. We will use ϕ to indicate the activation function used in the neural network. We will use $f : \mathbb{R}^{in} \to \mathbb{R}^{out}$ to refer to the function computed by the entire neural network. The computation performed by layer l of the network will be:

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

Network Jacobian Using the chain rule, the Jacobian of a neural network can be expanded as follows:

$$\frac{\partial f}{\partial \boldsymbol{x}} = \frac{\partial f}{\partial \boldsymbol{z}^L} \frac{\partial \boldsymbol{z}^L}{\partial \boldsymbol{h}^{L-1}} \frac{\partial \boldsymbol{h}^{L-1}}{\partial \boldsymbol{z}^{L-2}} \cdots \frac{\partial \boldsymbol{z}^2}{\partial \boldsymbol{h}^1} \frac{\partial \boldsymbol{h}^1}{\partial \boldsymbol{z}^1} \frac{\partial \boldsymbol{z}^1}{\partial \boldsymbol{x}} = \frac{\partial f}{\partial \boldsymbol{z}^L} \mathbf{W^{(L)}} \phi'(\mathbf{z^{(L-1)}}) \ldots \mathbf{W^{(2)}} \phi'(\mathbf{z^{(1)}}) \mathbf{W^{(1)}} \tag{1}$$

2.1 LIPSCHITZ FUNCTIONS

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

$$d_Y(f(x_1), f(x_2)) \leq L d_X(x_1, x_2)$$

where d_X and d_Y are metrics on X and 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_X and d_Y. Additionally, we take $X = \mathbb{R}^n$ and $Y = \mathbb{R}^m$ throughout. If the Lipschitz constant of a function is K, it is called a K-*Lipschitz* function. Equivalently, if the function is everywhere differentiable then its Lipschitz constant is given by the maximum of its Jacobian norm.

2.2 ENFORCING 1-LIPSCHITZ NEURAL NETWORKS

When we compose 1-Lipschitz functions we produce a 1-Lipschitz function. This means that to build a 1-Lipschitz neural network, it suffices to compose 1-Lipschitz linear transformations and activation functions.

1-Lipschitz Linear Transformations: To ensure that each linear map enforces the 1-Lipschitz constraint we need to restrict to maps that satisfy $||Wx||_p \leq ||x||_p$. This is exactly equivalent to restricting $||W||_p \leq 1$. For $p = 2$ this constraint is equivalent to ensuring that all singular values of W are less than or equal to 1. For $p = \infty$ this is equivalent to constraining W to satisfy,

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

Enforcing these constraints naively may be computationally expensive. Fortunately, techniques exist to efficiently ensure that $||W||_p = 1$ when $p = 2$ or $p = \infty$. We discuss these in more detail in Section 3.3.

1-Lipschitz Activation Functions: Most commonly used activation functions (such as ReLU (Krizhevsky et al., 2012), sigmoid, tanh, maxout (Goodfellow et al., 2013)) are already 1-Lipschitz, if they are scaled appropriately.

2.3 APPLICATIONS OF LIPSCHITZ NETWORKS

Wasserstein Distance Estimation Wasserstein-1 distance (also called Earth Mover Distance) provides a principled way to quantify the distance between two probability distributions and has found many applications in machine learning in recent years (Peyré et al., 2017; Genevay et al., 2017).

Using Kantorovich-Rubinstein duality (Villani, 2008), one can recast the Wasserstein distance estimation problem as a concave optimization problem, defined over 1-Lipschitz functions:

$$W(P_1, P_2) = \sup_{||f||_L < 1} \left(\mathbb{E}_{x\ P_1}[f(x)|] - \mathbb{E}_{x\ P_2}[f(x)|] \right) \tag{2}$$

Lipschitz constrained neural networks are often used to learn the function f. In this work, we explore the theoretical limitations of existing approaches and propose new techniques to close this gap.

Adversarial Robustness Adversarial examples are inputs to a machine learning system which have been designed to force misclassification Szegedy et al. (2013); Goodfellow et al. (2014). Formally, given a classifier f and a data point $\mathbf{x}$, we 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. Existing work has addressed the links between Lipschitz constraints and adversarial robustness (Tsuzuku et al., 2018; Raghunathan et al., 2018). Using a Lipschitz neural network one can provide robustness guarantees but existing approaches have both practical and theoretical limitations (Huster et al., 2018). In this work

Regularization Lipschitz constraints have also been explored as a regularization technique for deep neural networks. Sokolić et al. (2017) linked small Jacobian spectral norms to the generalization of neural networks. This is typically enforced by including a gradient penalty in the loss (Drucker & Le Cun, 1992) or restricting the spectral norm of each layer's weights (Yoshida & Miyato, 2017; Gouk et al., 2018).

3 METHODS

We begin by observing that 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. With this in mind, we focus on designing 1-Lipschitz neural networks with respect to the L_2 and L_∞ norms.

We will require each layer to have a Lipschitz constant of 1 which at first glance may seem overly strong as we only need the product of all layer's Lipschitz constants to be 1. However, if the activation functions preserve scalar multiplication (for example, $\text{ReLU}(a\mathbf{x}) = a\text{ReLU}(\mathbf{x})$) then these two settings are equivalent. Nonetheless, prior work suggests that this *atomic* construction is too restrictive and limits the class of functions which can be represented (Huster et al., 2018). In this work we show that in fact one can recover universal approximation even with this restriction.

3.1 JACOBIAN NORM PRESERVATION

We identify a key property 1-Lipschitz operations must have in order for them to compose expressive Lipschitz networks: they must be **Jacobian norm preserving**, which is defined as follows:

Definition 1. *Jacobian Norm Preservation: Let $f : \mathbb{R}^m \to \mathbb{R}^n$ ($m \geq n$) and $g : \mathbb{R}^n \to \mathbb{R}^k$ be two differentiable functions. f is Jacobian-norm preserving if and only if for any $\boldsymbol{x} \in \mathbb{R}^m$, the following property holds: $||\left(\frac{\partial g(f(\boldsymbol{x}))}{\partial f(\boldsymbol{x})}\right)_{i:} \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}|| = ||\left(\frac{\partial g(f(\boldsymbol{x}))}{\partial f(\boldsymbol{x})}\right)_{i:}||$, where the subscript $\left(\right)_{i:}$ denotes any row of $\frac{\partial g(f(\boldsymbol{x}))}{\partial f(\boldsymbol{x})}$.*

To describe why this property matters, we take the norm of the Jacobian expressed in Equation 1 and apply the Cauchy-Schwarz inequality for matrix multiplication on it:

$$||\frac{\partial f}{\partial \boldsymbol{x}}|| \leq ||\frac{\partial f}{\partial \boldsymbol{z}^L} \cdots \frac{\partial \boldsymbol{z}^{l_1}}{\partial \boldsymbol{h}^{l_1-1}}|| \cdot ||\frac{\partial \boldsymbol{h}^{l_1-1}}{\partial \boldsymbol{z}^{l_1-1}} \cdots \frac{\partial \boldsymbol{z}^{l_2}}{\partial \boldsymbol{h}^{l_2-1}}|| \cdot ||\frac{\partial \boldsymbol{z}^{l_2-1}}{\partial \boldsymbol{h}^{l_2-1}} \cdots \frac{\partial \boldsymbol{z}^1}{\partial \boldsymbol{x}}|| \tag{3}$$

The integers $1 \leq l_1 \leq l_2 \leq L$ denote where how we choose to partition the expanded Jacobian before applying the Cauchy Schwarz inequality.

Consider a scenario (necessary for universality) where we are trying to fit a piece-wise-linear function whose Jacobian norm is unity everywhere. Considering Equation 3 and the fact that the Jacobian of the entire network is bounded by 1 and,]any arbitrary contiguous chain of matrix multiplications in Equation 3 must have a norm of exactly 1:

$$||\frac{\partial \boldsymbol{z}^{l_1}}{\partial \boldsymbol{h}^{l_1-1}} \cdots \frac{\partial \boldsymbol{z}^{l_2}}{\partial \boldsymbol{h}^{l_2-1}}|| = 1 \quad , \quad l \in \{1...L\} \tag{4}$$

Algorithm 1: GroupSort activation

Inputs: $\boldsymbol{x} \in \mathbb{R}^n$, $g \in \mathbb{Z}_+$
for $i \in \{0 \dots g-1\}$ **do**
  $\boldsymbol{x}[i*g : (i+1)*g] = \text{SORT}(\boldsymbol{x}[i*g : (i+1)*g])$
end
return $\boldsymbol{x}$

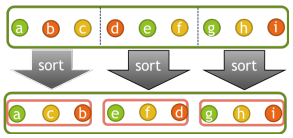

Figure 1: GroupSort activation. (green > yellow > red

Hence, neural network has cannot to lose Jacobian norm at any layer throughout the network for it to be capable of approximating functions whose Jacobian norm is 1 everywhere. To ensure this property, we propose to construct **Jacobian norm preserving** activations and linear transformations.

3.2 Norm-preserving activation functions

To achieve norm preservation, we propose a novel non-linear activation function which we call $\mathbf{GroupSort}$. This activation function takes a column vector $\boldsymbol{x} \in \mathbb{R}^n$, separates the elements into g groups, sorts each group into ascending order, and outputs the resulting "group sorted" vector. This is detailed in Figure 1 and Algorithm 1.

Differentiation: Applying $\mathbf{GroupSort}$ to a vector corresponds to conditionally shuffling the items in each group. This operation can be treated as multiplying the input vector with a permutation matrix, which can easily be differentiated through. Moreover, the Jacobian of this operation will be the permutation matrix meaning that $GroupSort$ is a Jacobian norm preserving activation.

MaxMin and FullSort: For ease of reference, when we pick a grouping size of 2 for GroupSort, we call the resulting activation **MaxMin**. When sort the *entire* input vector we call the resulting activation **FullSort**.

Connection between FullSort, MaxMin We can show the when given enough width and depth, FullSort and MaxMin activations have equal expressive power - they can compute all functions one other can compute. (see Appendix A for a qualitative explanation). Although FullSort can represent sorting operations much more compactly, we have found that optimization is more difficult with FullSort activation besides toy problems.

3.3 Norm-constrained linear maps

Enforcing $||W||_\infty = 1$ In all of our experiments, we use Algorithm 1 from Condat (2016) to project the weight matrices onto the L_∞ ball. Though the more sophisticated algorithms presented by the authors may be more efficient, we found that this technique worked well in our setting.

Enforcing $||W||_2 = 1$ There are several existing techniques proposed in the literature to achieve this. Spectral normalization (Miyato et al., 2018), employs an efficient implementation of power iteration which computes the spectral norm of each weight matrix. Each weight is then divided by its spectral norm to ensure the constraint. Parseval networks Cisse et al. (2017) add a regularization term, penalized by a hyperparameter β, which forces the weight matrices to be approximately orthonormal. In practice, they subtract the gradient of this term from the weights after each update which can be thought of as pushing the weights back towards the *Stiefel* manifold. The update can be written as $W = (1+\beta)W - \beta WW^TW$.

In fact, the Parseval update can be viewed as a special case of a more general algorithm which we here refer to as Bjorck Orthonormalization (Björck & Bowie, 1971). This is an iterative scheme which in its simplest form repeats exactly the Parseval update with $\beta = 0.5$. Importantly, this algorithm is fully differentiable and thus has a pullback operator for the Stiefel manifold. In practice we find that relatively few iterations are enough and we are even able to finetune the network with more iterations in its last few epochs to guarantee convergence.

In practice, the orthonormalization ensures Jacobian norm preservation and so despite being a subset of Spectral norm constrained weights we argue that this subset is the one that we should care about.

3.4 DIRECT JACOBIAN REGULARIZATION

Most existing work begins with the goal of constraining the spectral norm of the Jacobian and proceeds to achieve this by placing constraints on the weights of the network (Yoshida & Miyato, 2017). In this work, we propose a simple new technique which allows us to directly regularize the spectral norm of the Jacobian, $\sigma(J)$. This method differs from the ones described previously as the Lipschitz constant of the entire network is regularized using a single term, instead of at the layer level. We defer the full statement of this algorithm and related discussion to the appendix but refer to it throughout the rest of this paper as spectral Jacobian regularization.

4 RELATED WORK

Training Lipschitz neural networks has received both direct and indirect study as far back as Drucker & Le Cun (1992). In this work, the authors claimed that regularizing the Jacobian of neural networks improve their generalization performance. Using a Lipschitz constraint as a form of regularization has also been explored more recently (Gouk et al., 2018; Sokolić et al., 2017) with additional theoretical guarantees provided on the generalization performance.

Several proposals exist on the best way to train Lipschitz neural networks. Parseval networks (Cisse et al., 2017) are perhaps most similar to our work. Cisse et al. (2017) regularized the weights of the neural network to obey an orthogonal constraint and utilize Lipschitz activation functions. In fact, the corresponding update to the weights due to this regularization term is exactly one step of the Bjorck orthonormalization scheme. In Parseval networks the regularization can be thought of as projecting the weights closer to the space of orthogonal matrices after each update. This is a critical difference to our own work, in which a differentiable projection is used during each update.

Other regularization techniques penalize the Jacobian of the network which in turn constrains the Lipschitz constant (Gulrajani et al., 2017; Drucker & Le Cun, 1992; Gouk et al., 2018; Sokolić et al., 2017). While these methods have the advantage that it is typically easy to train neural networks under such constraints they do not provably enforce a Lipschitz constraint. In fact, in most cases the gradient will only be penalized close to the training data and no global guarantee can be made. In particular, Gulrajani et al. (2017) use a gradient penalty applied at randomly sampled points between two distributions but as shown by Gemici et al. (2018) this is not theoretically sound.

Huster et al. (2018) also explored universal approximation properties of Lipschitz networks. In their work, the authors prove that ReLU activations cannot be used to represent the absolute value function with L_∞ norm constraints. In this work we also show that ReLU activations are not sufficient in L_2 norm. However, we show that Lipschitz functions can be universally approximated with an L_∞ or L_2 norm if the correct activation function is used.

5 UNIVERSAL APPROXIMATION OF LIPSCHITZ FUNCTIONS

While universal approximation is well understood for general continuous functions, this is far from true when learning over a class of Lipschitz neural networks. In fact, existing work presents negative results that suggest this may be impossible to achieve using atomic constructions (Huster et al., 2018). Here we show that in fact we can recover universal approximation with an atomic construction and the GroupSort activation.

Huster et al. (2018) proved that ReLU networks with L_∞ norm constrained linear transformations cannot be universal Lipschitz function approximators. Here we extend this result to the L_2 norm by showing that no ReLU network is able to exactly produce the absolute value function. We then proceed to show that neural nets built with our proposed norm-preserving activation functions are universal Lipschitz function approximators in L_∞ norm.

Theorem 1. *For any finite integer width w and depth d, there does not exist a 1-Lipschitz ReLU networks (with respect to L_2 norm) of width w and depth d which can represent the absolute value function.*

Proof. (Proof by Contradiction) Assume such a network exists. We show in Appendix C.1 that if there must be a solution which has no neurons with 0 activations in its firsts layer. Using the chain rule, write the Jacobian of the network as,

$$\frac{\partial f}{\partial \boldsymbol{x}} = \frac{\partial f}{\partial \boldsymbol{h}^{(1)}} \frac{\partial \boldsymbol{h}^{(1)}}{\partial \boldsymbol{z}^{(1)}} \frac{\partial \boldsymbol{z}^{(1)}}{\partial \boldsymbol{x}} = \frac{\partial f}{\partial \boldsymbol{h}^{(1)}} \frac{\partial \mathbf{ReLU}(\mathbf{z}^{(1)})}{\partial \boldsymbol{z}^{(1)}} \mathbf{W}^{(1)}$$

We then apply the Cauchy-Schwarz inequality and use that the Jacobian norm of the absolute value function is 1 almost everywhere.

$$||\frac{\partial f}{\partial \boldsymbol{x}}||_2 = 1 \leq ||\frac{\partial f}{\partial \boldsymbol{h}^{(1)}}||_2 ||\frac{\partial \mathbf{ReLU}(\mathbf{z}^{(1)})}{\partial \boldsymbol{z}^{(1)}} \mathbf{W}^{(1)}||_{\mathbf{2}} \leq ||\frac{\partial \mathbf{f}}{\partial \mathbf{h}^{(1)}}||_{\mathbf{2}} ||\frac{\partial \mathbf{ReLU}(\mathbf{z}^{(1)})}{\partial \mathbf{z}^{(1)}}||_{\mathbf{2}} ||\mathbf{W}^{(1)}||_{\mathbf{2}}$$

This implies that all the Jacobian norms in the above expression must be equal to 1. In particular,

$$||\frac{\partial \mathbf{ReLU}(\mathbf{z}^{(1)})}{\partial \boldsymbol{z}^{(1)}} \mathbf{W}^{(1)}||_{\mathbf{2}} = \mathbf{1} \quad \text{and} \quad ||\mathbf{W}^{(1)}||_{\mathbf{2}} = \mathbf{1}$$

To conclude the proof, we evaluate the following quantity :

$$||\mathbf{W}^{\mathbf{1}}||_{\mathbf{2}}^{\mathbf{2}} - ||\frac{\partial \mathbf{ReLU}(\mathbf{z}^{(1)})}{\partial \mathbf{z}^{(1)}} \mathbf{W}^{(1)}||_{\mathbf{2}}^{\mathbf{2}} = \sum_{\mathbf{i=1}}^{\mathbf{n}} (\mathbf{1} - \frac{\partial \mathbf{ReLU}(\mathbf{z}^{(1)})}{\partial \mathbf{z}^{(1)}}_{\mathbf{ii}})^{\mathbf{2}} \mathbf{W}_{\mathbf{i}}^{(\mathbf{1})^{\mathbf{2}}} = \mathbf{0}$$

This indicates that $\frac{\partial \mathbf{ReLU}(\mathbf{z}^{\mathbf{1}})}{\partial \boldsymbol{z}^{\mathbf{1}}}$ must be the identity matrix. Hence, the first layer of the neural network must compute a linear function. We can then merge this linear computation into the second layer, and repeat the same argument until we collapse the entire neural network into a linear function. This is a contradiction, as the absolute value function is not linear.

Remark. *The same argument can be applied to any element-wise activation funcion whose element-wise derivatives can take values other than $+1$ or -1.*

This is a contradiction, we have shown that the only way the keep the output norm to be equal to 1 is to avoid applying any nonlinearity on it, as the Jacobian of a ReLU layer with off-units cannot be identity matrix. The above argument can be applied to any arbitrary ReLU layer, forcing the conclusion that the only way to preserve Jacobian norm is to avoid applying any nonlinearity in the entire neural network.

□

5.1 Universal Approximation for L_∞ norm

We now present theoretical guarantees on the approximation of Lipschitz functions with neural networks. To our knowledge, this is the first positive result showing that neural networks are capable of this.

We will first show that finding a class of neural networks which is able to approximate any 1-Lipschitz function is equivalent to showing that the class is closed under the maximum and minimum of representable functions. A similar result is presented in Lemma 4.1 in Yaacov (2010). We will then show that we are able to construct such a class of networks with the groupsort activation. We now proceed with the formal statements.

Definition 2. *Given a compact metric space (X, d_X) where d_X denotes the metric on X, we write $L_1(X, \mathbb{R})$ to denote the space of all Lipschitz-1 functions mapping X to $\mathbb{R}$ (with respect to L_∞ norm).*

Definition 3. *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 $L_1(X, \mathbb{R})$ with the property that for any two distinct elements $x, y \in X$ and any two real numbers a and b such that $||a - b||_\infty \leq d_X(x, y)$ there exists a function $f \in L$ such that $f(x) = a$ and $f(y) = b$. Then $L = L_1(X, \mathbb{R})$.*

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

The full proof of Lemma 1 is presented in the appendix. Note that Lemma 1 says that $\mathcal{A}$ is a universal approximator for Lipschitz-1 functions if and only if $\mathcal{A}$ is a lattice that separates points. Using Lemma 1, we can present the second of our key results. Huster et al. (2018) show that ReLU activations are unable to approximate $|x|$ but here we are able to present a much stronger result: atomically constrained networks with groupsort activations are able to represent any Lipschitz function in L_∞ norm.

Theorem 2. *Let LN_∞ denote the class of fully-connected neural networks with weight matrices that satisfy $||W||_\infty = 1$ and groupsort activations with group size 2. Let X be a closed and bounded subset of $\mathbb{R}^n$. Then the closure of LN_∞ is dense in $L_1(X, \mathbb{R})$.*

We refer readers to Appendix D for the formal proof of Theorem 2. We may also extend the restricted Stone-Weierstrass theorem in L_∞ norm to vector-valued functions, and consequently prove universal approximation in this setting. For scalar valued functions, we can modify the architecture to allow universal approximation in any L_p norms. Formally:

Lemma 2. *Consider LN_p, the set of neural networks in LN_∞ modified such that the first weight matrix satisfies $||W^{(1)}||_{p,\infty} = 1$. Then LN_p is dense in 1-Lipschitz functions with respect to the L_p norm.*

While these constructions rely on the L_∞ norm of the weight matrices being constrained, we find in practice that constraining the L_2 norm makes the networks easier to train and doesn't seem to violate the universal approximation results. However, we have yet to prove that a spectral norm constraint allows universal approximation in $L_1(X, \mathbb{R})$.

6 EXPERIMENTS

We are primarily interested in whether the theoretical limitations of activation functions which do not preserve the Jacobian norm are observed in practice. We first trained networks to solve simple Wasserstein distance estimation problems and found that norm preservation is critical. We extend these results to some standard classification problems and finally explore the benefits of Lipschitz constraints for robustness and interpretability.

6.1 PATHOLOGIES IN COMMON ACTIVATIONS

In these experiments we investigated whether the theoretical limitations of common activations we presented are of practical importance. To do this, we used the dual formulation of the Wasserstein-1 distance with some simple distributions which have closed form solutions. We trained various networks with 1-Lipschitz constraints and compared them to the ground-truth solutions.

6.1.1 PATHOLOGIES IN APPROXIMATING OPTIMAL DUAL SURFACES

In these experiments, we train neural networks to optimize the dual Wasserstein distance objective, expressed in Equation 2, between pairs of carefully chosen probability distributions whose optimal dual surface can be derived analytically. Appendix E.1 describes the distributions we chose and why they lead to the dual surfaces referred to below. We use the Bjorck procedure described in Section 3 to obtain strictly orthonormal weight matrices, although similar results could be obtained via Parseval training (Cisse et al., 2017) or spectral normalization (Miyato et al., 2018).

Approximating absolute value: Figure 2 shows the dual functions approximated by neural networks (3 hidden layers each with 128 hidden units) using the Tanh, ReLU, MaxOut, Maxmin and FullSort. It can be seen that non-Jacobian norm preserving activations are incapable of approximating the absolute value function. While we observed that increasing the network depth helps ReLU and MaxOut activations, the representational bottleneck showcased in Figure 2 leads to limitations as the problem dimensionality increases.

Approximating multiple 2D cones: The surfaces approximated by neural networks (3 hidden layers with with 312 units) using ReLU, MaxOut, MaxMin and FullSort (Figure 3) showcase an even more serious pathology with non-Jacobian norm preserving activations. The surfaces learned by such activations lack enough slope to properly optimize the Wasserstein dual problem. While there are use-cases of Lipschitz neural networks which are invariant to an arbitrary scaling of the network output, such as when training the critic network in a WGAN (Arjovsky et al., 2017), this experiment shows that capacity issues related to common activation functions such as ReLU and MaxOut cause surface deformities that **cannot** be fixed by arbitrary scaling.

Approximating high dimensional circular cones: We evaluated the performance of architectures built with different activation functions as we increase the dimensionality of the support of the probability distributions. As shown in Table 1, this leads to significant drops in the expressivity of Lipschitz networks built with non-Jacobian norm preserving activations, and increasing the depth of

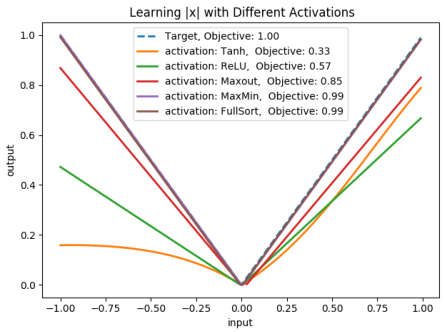

Figure 2: Approximating the absolute value function via dual functions. The objective values indicate the Wasserstein Distance estimated by each network.

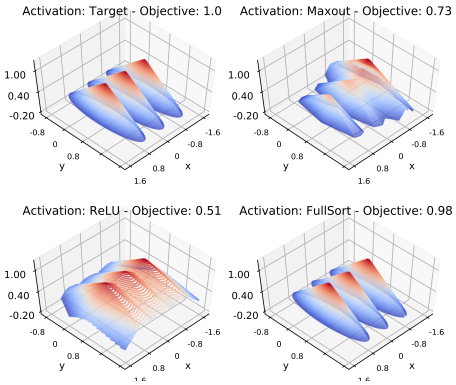

Figure 3: Approximating three circular cones via dual functions. The objective values represent the Wasserstein distance estimated by each networks.

the network did not lead to considerable improvements. We also observed that while the MaxMin activation performs significantly better, it also needs network depth in order to converge to the optimal solution. Surprisingly, the FullSort activation has no difficulty approximating high dimensional circular cones, even with only two hidden layers.

6.2 Practical ReLU Pathologies

Activ.	**Input Dim=128**		**Input Dim=256**	
Depth	*3*	*7*	*3*	*7*
ReLU	0.51	0.60	0.50	0.53
Maxout	0.66	0.71	0.60	0.66
MaxMin	0.87	0.95	0.83	0.93
FullSort	**0.99**	**0.99**	**0.99**	**0.99**

Table 1: **Effect of problem dimensionality on expressiveness:** Testing how well different activation functions and depths can optimize the dual Wasserstein objective with different input dimensionality and depths. The optimal dual surface is expected to obtain a Wasserstein Distance of unity.

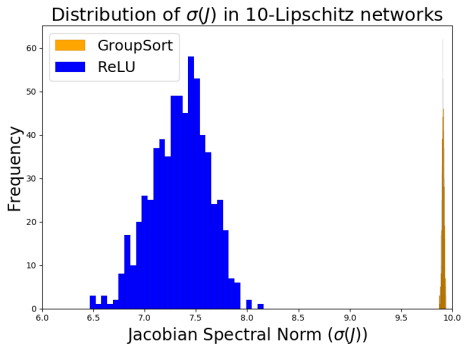

Figure 4: **Jacobian spectral norm distribution over training set** We compare the Jacobian spectral norm of ReLU and GroupSort networks.

Thus far we have focused on examples where the gradient of the network should be 1 almost everywhere. But for many practical tasks we do not need to meet this strong condition. Then we should ask, are these pathologies relevant in other settings?

How much of the Lipschitz capacity can we use? We have shown theoretically that ReLU networks approach linear networks as they utilize the full gradient capacity allowed to them with the Lipschitz constraints. To understand these implications practically, we trained two MNIST classifiers with orthonormal weight constraints enforced to ensure that they are 10-Lipschitz functions. We then looked at the distribution of the spectral radius of the Jacobian over the training data. Figure 4 displays this distribution for each networks. We observed that while both networks satisfy the expected 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.

Model	ReLU	Maxout	Maxmin	GroupSort(4)	GroupSort(9)
MNIST	1.65	2.32	2.57	**2.73**	2.69
CIFAR10	3.00	4.02	4.38	4.54	**4.59**

Table 2: Estimating the Wasserstein Distance between the data and generator distributions using 1-Lipschitz neural network, for an MNIST (WGAN (Arjovsky et al., 2017)) and a CIFAR10 (Improved WGAN) (Gulrajani et al., 2017).

(a) Trained on MNIST. (b) Trained on CIFAR10.

Figure 5: The above images are sampled from the wasserstein GANs whose critic architectures were built using Jacobian norm preserving atomic units.

6.3 Wasserstein Distance Estimation

We now estimate the Wasserstein Distance between two high dimensional probability distributions through samples. Specifically, we try to compute a lower bound on the Wasserstein distance between the generator distribution of a GAN, and the data distribution it was trained on.

We want to emphasize that unlike training critic networks for Wasserstein GANs, where imposing a K-Lipschitz constraint is sufficient, the Wasserstein distance estimation problem requires one to search over *exactly* 1-Lipschitz neural networks. This is a much stricter constraint. Importantly, dual surfaces that optimize the dual Wasserstein distance formulations contain many subdomains in which the Jacobian norm is equal to 1 (for a rigorous statement see Corollary 1 in Gemici et al. (2018)). This is a scenario in which, as explained in Section 3.1, Jacobian norm preservation is a key property we want our neural network architecture to have.

6.3.1 Lower Bounds on MNIST and CIFAR10 GANS

In this experiment, We first trained a GAN variant on MNIST and CIFAR10 datasets, then froze the weights of the generator. Using samples from the generator and original data distribution, we trained several independent 1-Lipschitz neural networks. We used a shallow fully connected architecture (3 layers with 720 neurons in hidden layers). As can be seen in Table 2, using norm-preserving activation functions helps achieve a significantly tighter lower bound on Wasserstein distance between the data and generator distributions for both MNIST and CIFAR10 experiments.

Training WGANs We show that we are also able to train Wasserstein GANs using our proposed 1-Lipschitz activations and linear transformations. We borrow the discriminator and generator architectures directly from Chen et al. (2016), but replace the standard convolutional and fully connected layers with their Bjorck counterparts. We also drop the batch normalization layers, as it goes against our overall design principle of having explicit control over the exact Lipschitz constant of the network. Figure 5 shows some MNIST and CIFAR10 samples generated using our GAN variant. We leave the analysis and evaluation of the GANs built with our techniques to a future study.

6.4 Classification

We compare a wide range of Lipschitz architectures and training schemes on some simple benchmark classification tasks. We demonstrate that we are able to learn Lipschitz neural networks which are expressive enough to perform classification without sacrificing performance.

	ReLU	MaxMin	GroupSort-4	FullSort	Maxout
Standard	1.61	1.47	1.62	3.53	1.40
Dropout	1.52	1.48	1.74	2.72	1.38
Bjorck	1.54	1.25	1.43	2.06	1.43
Spectral Norm	1.54	1.26	1.32	2.94	1.26
Spectral Jac	1.05	1.09	1.24	1.93	1.02
Parseval	1.43	1.40	1.44	3.36	1.35
L_∞	2.25	2.28	2.22	4.88	1.98

Table 3: **MNIST classification** Test error shown for different architectures and activation functions.

Training Size	Standard		Parseval ($\beta = 0.01$)		Bjorck (1-Lipschitz)	
	ReLU	GroupSort	ReLU	GroupSort	ReLU	GroupSort
300	35.77	31.04	19.23	22.03	**15.32**	15.66
500	15.66	14.09	16.14	14.78	13.4	**11.08**
1000	14.92	13.40	8.72	9.57	10.64	**9.56**
5000	5.97	4.75	4.06	**3.97**	6.76	5.04
10000	3.98	3.48	2.87	**2.79**	5.64	3.96

Table 4: **MNIST Classification with limited training data** Test error for varying architectures and activations per training data size.

MNIST Classification We explored classification with a 3-layer fully connected network with 1024 hidden units in each layer. Each model was trained with the Adam optimizer (Kingma & Ba, 2014). The full results are presented in Table. 3.

For all models the GroupSort activation is able to perform classification well - especially when the Lipschitz constraint is enforced. Surprisingly, we found that we could even apply the GroupSort activation to sort the entire hidden layer and still achieve reasonable classification performance. When aiming to train good classifiers we found that spectral Jacobian regularization was most effective.

While the Parseval networks are capable of learning a strict Lipschitz constraint this does not always true in practice. A small beta value leads to slow convergence towards orthonormal weights. When early stopping is used, which is typically important to achieve good validation accuracy, it is difficult to ensure that the resulting network is indeed 1-Lipschitz.

Classification with little data While enforcing the Lipschitz constraint aggressively could hurt overall predictive performance, it decreases the generalization gap substantially. Motivated by the observations of Bruna & Mallat (2013) we investigated the performance of Lipschitz networks on small amounts of training data, where learning robust features to avoid overfitting is critical.

For these experiments we kept the same network architecture as before. We trained standard unregularized networks, 1-Lipschitz neural networks enforced with the Bjorck algorithm, and Parseval networks while varying the number of training images. Despite the fact that the neural networks are extremely overparametrized, we found that using Lipschitz constraints we were able to significantly improve performance, as presented in Table 4.

Classification on CIFAR We briefly explored classification of CIFAR-10 with a Resnet32 to confirm that the GroupSort activation can be effective even in this more challenging regime. We did not tune any method extensively and present these results primarily to show that GroupSort is viable outside of MNIST classification. We hope to explore this further in future work.

	Standard		Parseval		Spec Jac Regularization	
	ReLU	GroupSort	ReLU	GroupSort	ReLU	GroupSort
CIFAR-10	89.90	89.92	88.08	89.88	90.51	90.02

Table 5: **CIFAR Classification** Test acuracy for varying architectures and activations.

(a) Standard network (b) Lipschitz network

Figure 6: Gradients of input images with respect to targeted cross-entropy loss. One image from each class was chosen randomly from the test set. The first row shows the original images with following rows showing the gradient with different class targets (0-9).

	Standard	1-Lipschitz	0.1-Lipschitz
FGS	0.013	0.031	**0.033**
PGD	0.003	0.018	**0.021**

Table 6: **Adversarial robustness** The minimum squared distance required to force misclassification as the Lipschitz constant increases.

6.5 Robustness and Interpretability of Lipschitz Networks

We explored the robustness of Lipschitz neural networks to adversarial examples. We trained networks with varying Lipschitz constants and attacked them using the FGS and PGD methods (Szegedy et al., 2013; Madry et al., 2017). To test the claim that Lipschitz networks are able to correctly classify adversarial examples produced with larger perturbations we estimated the minimum squared distance required to force misclassification and reported the average over 2000 points chosen randomly from the test set. As expected, a smaller Lipschitz constant allows larger perturbations to be classified successfully.

Tsipras et al. (2018) reported that networks trained using adversarial training learn robust features which allow them to have interpretable gradients. We found that the same is true for Lipschitz networks, even without using adversarial training. The gradients with respect to the inputs are displayed for a standard network and a Lipschitz network in Figure 6.

7 Conclusion

We have identified Jacobian norm preservation as a critical component of Lipschitz network design and showed that failure to achieve this leads to pathological networks in practice. Additionally, we made significant steps towards understanding the requirements of universal Lipschitz function approximators and presented classes of networks which are able to approximate arbitrary Lipschitz functions.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

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

Joan Bruna and Stephane Mallat. 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`.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pp. 2172–2180, 2016.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. *arXiv preprint arXiv:1704.08847*, 2017.

Laurent Condat. Fast projection onto the simplex and the $\boldsymbol{l_1}$ ball. *Mathematical Programming*, 158 (1-2):575–585, 2016.

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

Mevlana Gemici, Zeynep Akata, and Max Welling. Primal-dual wasserstein gan. *arXiv preprint arXiv:1805.09575*, 2018.

Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. *arXiv preprint arXiv:1706.00292*, 2017.

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In Sanjoy Dasgupta and David McAllester (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`.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael Cree. Regularisation of neural networks by enforcing lipschitz continuity. *arXiv preprint arXiv:1804.04368*, 2018.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

Todd Huster, Cho-Yu Jason Chiang, and Ritu Chadha. Limitations of the lipschitz constant as a defense against adversarial examples. *arXiv preprint arXiv:1807.09705*, 2018.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 12 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. Technical report, 2017.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. There is no free lunch in adversarial robustness (but there are unexpected benefits). *arXiv preprint arXiv:1805.12152*, 2018.

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

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

Itaï Ben Yaacov. Lipschitz functions on topometric spaces. *arXiv preprint arXiv:1010.1600*, 2010.

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

Appendices

A MaxMin, FullSort

A.1 FullSort and MaxMin

Representing MaxMin with FullSort: The MaxMin operation can be represented with a FullSort operation, sandwiched between two identity transformations (which are 1-Lipschitz) with adjustable biases. This directly follows from the observation that we can adjust the biases of the linear transformation preceding a FullSort operation such that the units are "chunked" in groups of two. Sorting this entire chunked pre-activation vector then removing the effect of the biases in the following identity transformation is equivalent to applying MaxMin directly on the activation.

Representing FullSort with MaxMin: It is possible to implement bubble sort with a neural net with MaxMin activations. This requires $\frac{n*(n+1)}{1}$ layers whose weight matrices are appropriate permutation matrices (hence 1-Lipschitz). In the first n layers, we pick the appropriate permutation matrices and biases to sweep the input vector bottom-up with MaxMins, so that we are guaranteed to have the largest (or smallest) value in the matrix on the top of the matrix. We then go over the same procedure repeatedly to sort the entire input vector.

B Spectral Jacobian Regularization

In this section we present details on the algorithm used to regularize the spectral norm of the network Jacobian. The intuition follows that of Yoshida & Miyato (2017), who apply power iteration to estimate the singular values of the weight matrices online. The authors also discuss computing the spectral radius of the Jacobian directly, and related quantities such as the Frobenius norm, but dismiss this as being too computationally expensive.

Power iteration can be used to compute the leading singular value of a matrix J with the following repeated steps,

$$\mathbf{v}_k = J^T \mathbf{u}_{k-1} / ||J^T \mathbf{u}_{k-1}||_2, \mathbf{u}_k = J\mathbf{v}_k / ||J\mathbf{v}_k||_2$$

Then we have $\sigma(J) \approx \mathbf{u}^T J \mathbf{v}$. There are two challenges that must be overcome to implement this in practice. First, the algorithm requires higher order derivatives which leads to increased computational overhead. However, the tradeoff is often reasonable in practice, see e.g. Drucker & Le Cun (1992). Second, the algorithm requires both Vector-Jacobian products and Jacobian-Vector products. The former can be computed with reverse-mode automatic differentiation but the latter requires the less common forward-mode. Fortunately, one can recover forward-mode from reverse mode by constructing Vector-Jacobian products and utilizing the transpose operator. In this setting, we can actually re-use the intermediate reverse-mode backpropagation within the algorithm which further reduces the computational overhead. The algorithm itself is presented as Algorithm 2.

We present this algorithm primarily to be used for regularization but this could also be used to enforce the Lipschitz constraint by rescaling the output of the entire network by the estimate of the Jacobian spectral norm in a similar fashion to weight spectral normalization Miyato et al. (2018).

C Insufficiency of ReLU Proofs

C.1 Pushing 0-neurons to following layers

: Lets say there is a neuron in the first layer $\mathbf{W}^{(\mathbf{1})}$ of a neural network with all 0 weights. We can construct another neural network where we add $W^{(2)}_{\cdot j} \cdot b^{(1)}_j$ to the biases of the second layer, and completely get rid of the 0 neurons in the first layer without having any effect on the function that is being computed by the neural network.

Algorithm 2: Spectral Jacobian Regularization

Initialize u randomly, choose hyperparameter $\lambda > 0$
for *data batch (X,Y)* **do**
  Compute logits $f_\theta(X)$
  Compute loss $\mathcal{L}(f_\theta(X), Y)$
  Compute $\mathbf{g} = \mathbf{u}^T \frac{\partial \mathbf{f}}{\partial \mathbf{x}}$, using reverse mode
  Set $\mathbf{v} = \mathbf{g}/||\mathbf{g}||_2$
  Compute $\mathbf{h} = (\mathbf{v}^T \frac{\partial \mathbf{g}}{\partial \mathbf{u}})^T = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{v}$, using reverse mode
  Update $u = \mathbf{h}/||\mathbf{h}||_2$
  Compute parameter update from $\frac{\partial}{\partial \theta}\left(\mathcal{L} + \lambda \mathbf{u}^T \mathbf{h}\right)$
end

D UNIVERSAL APPROXIMATION PROOFS

D.1 UNIVERSAL APPROXIMATION OF 1-LIPSCHITZ FUNCTIONS

Here we present formal proofs related to finding neural network architectures which are able to approximate any 1-Lipschitz function. We begin with a proof of Lemma 1.

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 $L_1(X, \mathbb{R})$ with the property that for any two distinct elements $x, y \in X$ and any two real numbers a and b such that $||a - b||_\infty \leq d_X(x, y)$ there exists a function $f \in L$ such that $f(x) = a$ and $f(y) = b$. Then $L = L_1(X, \mathbb{R})$.*

Proof. We aim to show that for any $g \in L_1(X, \mathbb{R})$ and $\epsilon > 0$ we can find $f \in L$ such that $||g - f||_\infty < \epsilon$ (i.e. the largest difference is ϵ).

Fix $x \in X$. Then for each $y \in X$, we have an $f_y \in L$ with $f_y(x) = g(x)$ and $f_y(y) = g(y)$. This follows from the separation property of L and, using the fact that g is Lipschitz-1, $||g(x) - g(y)||_2 \leq d_X(x, y)$.

Define $V_y = \{z \in X : f_y(z) < g(z) + \epsilon\}$. Then V_y is open and we have $x, y \in V_y$. Therefore, the collection of sets $\{V_y\}_{y \in X}$ is an open cover of X. By the compactness of X, there exists some finite subcover of X, say, $\{V_{y_1}, \ldots, V_{y_n}\}$, with corresponding functions $f_{y_1}, \ldots, f_{y_n}$.

Let $F_x = min(f_{y_1}, \ldots, f_{y_n})$. Since L is a lattice we must have $F_x \in L$. And moreover, we have that $F_x(x) = g(x)$ and $F_x(z) < g(z) + \epsilon$, for all $z \in X$.

Now, define $W_x = \{z \in X : F_x(z) > g(z) - \epsilon\}$. Then W_x is an open set containing x. Therefore, the collection $\{W_x\}_{x \in X}$ is an open cover of X and admits a finite subcover, $\{W_{x_1}, \ldots, W_{x_m}\}$, with corresponding functions $F_{x_1}, \ldots, F_{x_m}$.

Let $G = max(F_{x_1}, \ldots, F_{x_m}) \in L$. We have $G(z) > g(z) - \epsilon$, for all $z \in X$.

Combining both inequalities, we have that $g(z) - \epsilon < G(z) < g(z) + \epsilon$, for all $z \in X$. Or more succinctly, $||g - G||_\infty < \epsilon$. The theorem is proved by taking $f = G$. □

We now proceed to prove Theorem 2.

Theorem 2. *Let LN_∞ denote the class of fully-connected neural networks with weight matrices that satisfy $||W||_\infty = 1$ and groupsort activations with group size 2. Let X be a closed and bounded subset of $\mathbb{R}^n$. Then the closure of LN_∞ is dense in $L_1(X, \mathbb{R})$.*

Proof. The first property we require is separation of points. This follows trivially as given four points satisfying the required conditions we can find a linear map with the required L_∞ matrix norm that fits them. It remains then to prove that we can construct a lattice under this constraint. We begin by considering two 1-Lipschitz neural networks, f and g. We wish to design an architecture which is guaranteed to be 1-Lipschitz and can represent both $max(f, g)$ and $min(f, g)$.

The key insight we will use is the idea that we can split the network into two parallel *channels* which each computes one of f and g. At the end of the network, we can then select one of these channels depending on whether we want the max or the min.

Each of the networks f and g is determined by a set of weights and biases, we will denote these $[W_f^{(1)}, b_f^{(1)}, \ldots, W_f^{(n)}, b_f^{(n)}]$ and $[W_g^{(1)}, b_g^{(1)}, \ldots, W_g^{(n)}, b_g^{(1)}]$ for f and g respectively. For now, assume that these networks are of equal depth (we can lift this assumption later) however we make no assumptions on the width. We will now construct $h = max(f, g)$ in the form of a 1-Lipschitz neural network. To achieve this, we will design a network h which first concatenates the first layers of networks f and g and then computes f and g separately before combining them at the end.

We take the first weight matrix of h to be $W_h^{(1)} = [W_f^{(1)} \; W_g^{(1)}]^T$, that is the weight matrices of f and g stacked vertically. This matrix necessarily satisfies $||W_h^{(1)}||_\infty = 1$. Similarly, the bias will be those from the first layers of f and g stacked vertically. Then the first layer's pre-activations will be exactly the pre-activations of f and g stacked vertically.

For the following layers, we construct the biases in the same manner (vertical stacking). We construct the weights by constructing new block-diagonal weight matrices. That is, given $W_f^{(i)}$ and $W_g^{(i)}$, we take

$$W_h^{(i)} = \begin{bmatrix} W_f^{(i)} & 0 \\ 0 & W_g^{(i)} \end{bmatrix}$$

This matrix also has L_∞ norm equal to 1. We repeat this for each of the layers in f and g and end up with a final layer which has two units, f and g. We can then take maxmin of this final layer and take the inner product with $[1, 0]$ to recover the max ($[0, 1]$ would give the min).

Finally, we must address the case where the depth of f and g are different. In this case we notice that we are able to represent the identity function with maxmin activations. To do so observe that after the pre-activations have been sorted we can multiply by the identity and the sorting activation afterwards will have no additional effect. Therefore, for the channel that has the smallest depth we can add in these additional identity layers to match the depths and resort to the above case.

We have shown that the set of neural networks is a lattice which separates points, and thus by Lemma 1 it must be dense in $L_1(X, \mathbb{R})$. □

Note that we could have also used the maxout Goodfellow et al. (2013) to complete this proof. This makes sense, as the maxout activation is also norm-preserving in L_∞. However, this does not hold when using a spectral norm constraint on the weights. We now present several consequences of the theoretical results given above.

This result can be extended easily to vector-valued functions for the L_∞ norm by noticing that the space of such 1-Lipschitz functions is a lattice. We may apply the Stone-Weierstrass proof to each of the coordinate functions independently and use the same construction as in Theorem 2 modifying only the last layer which will now reorder the outputs of each function to do a pairwise comparison and then select the relevant components to produce the max or the min.

We may also extend this result to any L_p norm:

Lemma 2. *Consider LN_p, the set of neural networks in LN_∞ modified such that the first weight matrix satisfies $||W^{(1)}||_{p,\infty} = 1$. Then LN_p is dense in 1-Lipschitz functions with respect to the L_p norm.*

The proof follows the same construction as Theorem 2 with the modified first layer. The result follows directly by considering the Lipschitz constant of the atomic construction together with the definition of the mixed matrix p-norm.

E Experiment Details

Here we present additional details of the experiments conducted in the main paper.

E.1 SIMPLE PROBABILITY DISTRIBUTIONS AND THEIR CORRESPONDING DUAL SURFACES

Absolute value: We pick $p_1(\text{x}) = \delta_0(x)$ and $p_2(\text{x}) = \frac{1}{2}\delta_{-1}(x) + \frac{1}{2}\delta_1(x)$, where $\delta_\alpha(x)$ stands for the Dirac delta function located at α. It can be shown that the optimal dual surface learned while computing the Wasserstein distance between p_1 and p_2 is the absolute value function. This also makes intuitive sense, as the function that assigns "as low values as possible" at $x = 0$ and assigns "as low values as possible" at $x = -1$ and $x = 1$ while making sure that the absolute value of the slope of the function never exceeds 1, must be the absolute value function.

The Wasserstein distance obtained using absolute value as the dual function is 1. This becomes clearer when viewed from the primal problem, as the transport plan that will minimize the primal objective will simply be to map the center Dirac delta equally to the ones near it. This requires all the unit masses to be moved by a distance of 1.

Multiple 2D Circular Cones: We describe the probability distributions p_1 and p_2 implicitly by describing how we sample from them. p_1 is sampled from by selecting one of the three points ($(-2, 0)$, $(0, 0)$ and $(2, 0)$) uniformly. p_1 is sampled from by first uniformly selecting one of the three points aforementioned, then uniformly selecting a point on the circle surrounding it, with radius 1. Hence Wasserstein dual problem aims to find a Lipschitz function which assigns "as high as possible" values to the three points, and "as low as possible" values to the circles with radius 1 surrounding the three points. Hence, the optimal dual function must consist of three cones centered around $(-2, 0)$, $(0, 0)$ and $(2, 0)$. The behavior of the function outside this support doesn't have an impact on the solution.

The Wasserstein distance between p_1 and p_2 is equal to 1. . From the perspective of the primal formulation, the optimal transport plan must simply consist of mapping the probability mass to the nearby circles surrounding them uniformly. This leads to an expected transport (Wasserstein distance) cost of 1.0.

n Dimensional Circular Cones: This is a simple extension of the absolute value case described above.

Here, we check how the performance of architectures built with different activation functions as we increase input dimensionality $n^{(in)}$. We pick p_1 as the Dirac delta function located at the origin, and sample from p_2 by uniformly selecting a point from the $n^{(in)}$ dimensional spherical shell with radius 1, centered at the origin. Following similar arguments developed for absolute value and multiple 2D cones, it can be shown that the optimal dual function is a single $n^{(in)}$ dimensional circular cone and the Wasserstein distance is also equal to unity.

E.2 CLASSIFICATION

For the MNIST classification task we search of the hyperparameters are follows. For the Bjorck, L_∞ constrained, and Spectral Norm architectures we try networks with a guaranteed Lipschitz constant of 0.1, 1, 10 or 100. For Parseval networks we tried β values in the range 0.001, 0.01, 0.1, 0.5. For spectral Jacobian regularization we scaled the penalty by 0.01, 0.05, or 0.1.