Approximating Lipschitz continuous functions with GroupSort neural networks

Ugo Tanielian, Maxime Sangnier, Gerard Biau

Approximating Lipschitz continuous functions with GroupSort neural networks

U. Tanielian
Criteo, Sorbonne Université
u.tanielian@criteo.com
Paris, France

M. Sangnier
Sorbonne Université
maxime.sangnier@upmc.fr
Paris, France

G. Biau
Sorbonne Université
gerard.biau@upmc.fr
Paris, France

Abstract

Recent advances in adversarial attacks and Wasserstein GANs have advocated for use of neural networks with restricted Lipschitz constants. Motivated by these observations, we study the recently introduced GroupSort neural networks, with constraints on the weights, and make a theoretical step towards a better understanding of their expressive power. We show in particular how these networks can represent any Lipschitz continuous piecewise linear functions. We also prove that they are well-suited for approximating Lipschitz continuous functions and exhibit upper bounds on both the depth and size. To conclude, the efficiency of GroupSort networks compared with more standard ReLU networks is illustrated in a set of synthetic experiments.

1 Introduction

In the past few years, developments in deep learning have highlighted the benefits of operating neural networks with restricted Lipschitz constants. An important illustration is provided by robust machine learning, where networks with large Lipschitz constants are prone to be more sensitive to adversarial attacks, in the sense that small perturbations of the inputs can lead to significant misclassification errors (e.g., Goodfellow et al., 2015). In order to circumvent these limitations, Gao et al. (2017), Esfahani and Kuhn (2018), and Blanchet et al. (2019) studied a new regularization scheme based on penalizing the gradients of the networks. Constrained neural networks also play a key role in the different but not less important domain of Wasserstein GANs (Arjovsky et al., 2017), which take advantage of the dual form of the 1-Wasserstein distance expressed as a supremum over the set of 1-Lipschitz functions (Villani, 2008). This formulation has been shown to bring training stability and is empirically efficient (Gulrajani et al., 2017). In this context, many different ways have been explored to restrict the Lipschitz constants of the discriminator. One possibility is to clip their weights, as advocated by Arjovsky et al. (2017). Other solutions involve enforcing a gradient penalty (Gulrajani et al., 2017) or penalizing norms of the matrices of the weights (Miyato et al., 2018).

However, all of these operations are delicate and may significantly affect the expressive power of the neural networks. For example, Huster et al. (2018) show that ReLU neural networks with constraints on the weights cannot represent even the simplest functions, such as the absolute value. In fact, little is known regarding the expressive power of such restricted networks, since most studies interested in the expressiveness of neural networks (e.g., Hornik et al., 1989; Cybenko, 1989; Raghu et al., 2017) do not take into account eventual constraints on their architectures. As far as we know, the most recent attempt to tackle this issue is by Anil et al. (2019). These authors exhibit a family of neural networks, with constraints on the weights, which is dense in the set of Lipschitz continuous functions on a compact set. To show this result, Anil et al. (2019) make critical use of GroupSort activations.

Motivated by the above, our objective in the present article is to make a step towards a better mathematical understanding of the approximation properties of Lipschitz feedforward neural networks using GroupSort activations. Our contributions are threefold:

- (i) We show that GroupSort neural networks, with constraints on the weights, can represent any Lipschitz continuous piecewise linear function and exhibit upper bounds on both their depth and size. We make a connection with the literature on the depth and size of ReLU networks (in particular Arora et al., 2018; He et al., 2018).
- (ii) Building on the work of Anil et al. (2019), we offer

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

upper bounds on the depth and size of GroupSort neural networks that approximate 1-Lipschitz continuous functions on compact sets. We also show that increasing the grouping size may significantly improve the expressivity of GroupSort networks.

(*iii*) We empirically compare the performances of GroupSort and ReLU networks in the context of function regression estimation and Wasserstein distance approximation.

The mathematical framework together with the necessary notation is provided in Section 2. Section 3 is devoted to the problem of representing Lipschitz continuous functions with GroupSort networks of grouping size 2. The extension to any arbitrary grouping size is discussed in Section 4 and numerical illustrations are given in Section 5. For the sake of clarity, all proofs are gathered in the Appendix.

2 Mathematical context

We introduce in this section the mathematical context of the article and describe more specifically the GroupSort neural networks, which, as we will see, play a key role in representing and approximating Lipschitz continuous functions.

Throughout the paper, the ambient space $\mathbb{R}^d$ is assumed to be equipped with the Euclidean norm $\|\cdot\|$. For E a subset of $\mathbb{R}^d$, we denote by $\mathrm{Lip}_1(E)$ the set of 1-Lipschitz real-valued functions on E, i.e.,

$$\mathrm{Lip}_1(E) = \left\{ f : E \to \mathbb{R} : |f(x) - f(y)| \leqslant \|x - y\|,\ (x,y) \in E^2 \right\}$$

Let $k \geqslant 2$ be an integer. We let $\mathscr{D}_k = \{D_{k,\alpha} : \alpha \in \Lambda\}$ be the class of functions from $\mathbb{R}^d$ to $\mathbb{R}$ parameterized by feedforward neural networks of the form

$$D_{k,\alpha}(x) = \underset{1\times v_{q-1}}{V_q}\ \sigma_k\Big(\underset{v_{q-1}\times v_{q-2}}{V_{q-1}}\ \cdots \sigma_k\Big(\underset{v_2\times v_1}{V_2}\ \sigma_k\Big(\underset{v_1\times D}{V_1}\, x + \underset{v_1\times 1}{c_1}\Big) + \underset{v_2\times 1}{c_2}\Big) + \underset{v_{q-1}\times 1}{c_{q-1}}\Big) + \underset{1\times 1}{c_q}\,, \tag{1}$$

where $q \geqslant 2$ and the characters below the matrices indicate their dimensions (lines $\times$ columns). For $q = 1$, we simply let $D_{k,\alpha}(x) = V_1 x + c_1$ be a simple linear regression in $\mathbb{R}$ without hidden layers. Thus, a network in $\mathscr{D}_k$ has $(q-1)$ hidden layers, and hidden layers from depth 1 to $(q-1)$ are assumed to be of respective widths v_i, $i = 1,\ldots,q-1$, *divisible by* k. Such a network is said to be of depth q and of size $v_1 + \cdots + v_{q-1}$. The matrices V_i are the matrices of weights between layer i and layer $(i+1)$ and the c_i's are the corresponding offset vectors (in column format). So, altogether, the vectors $\alpha = (V_1,\ldots,V_q,c_1,\ldots,c_q)$ represent the parameter space Λ of the functions in $\mathscr{D}_k$. With respect to the activation functions σ_k, we propose to use the GroupSort activation, which separates the pre-activations into groups and then sorts each group into ascending order.

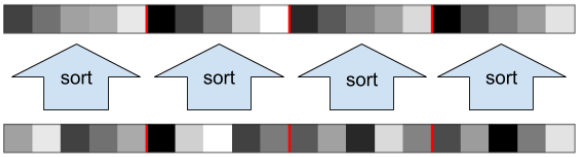


Figure 1: GroupSort activation with a grouping size 5. Source: Anil et al. (2019).

The GroupSort function splits the input into n different groups of k elements each: $G_1 = \{x_1,\ldots,x_k\},\ldots,G_n = \{x_{nk-(k-1)},\ldots,x_{nk}\}$, and then orders each group by decreasing order. Thus, the GroupSort function with a grouping size $k \geqslant 2$ is applied on a given vector $(x_1,\ldots,x_{kn})$ as follows:

$$\sigma_k(x_1,\ldots,x_k,\ldots,x_{nk-(k-1)},\ldots,x_{nk}) = \left(x_{(k)}^{G_1},\ldots,x_{(1)}^{G_1},\ldots,x_{(k)}^{G_n},\ldots,x_{(1)}^{G_n}\right),$$

where $(x_{(k)}^{G_i},\ldots,x_{(1)}^{G_i})$ corresponds to the decreasing ordering in the group G_i.

This activation is applied on groups of k components, which makes sense in (1) since the widths of the hidden layers are assumed to be divisible by k. GroupSort has been introduced in Anil et al. (2019) as a 1-Lipschitz activation function that preserves the gradient norm of the input. An example with a grouping size $k = 5$ is given in Figure 1. With a slight abuse of vocabulary, we call a neural network of the form (1) a GroupSort neural network. We note that the GroupSort activation can recover the standard rectifier function. For example, $\sigma_2(x,0) = (\mathrm{ReLU}(x), -\mathrm{ReLU}(-x))$, but the converse is not true.

Throughout the manuscript, the notation $\|\cdot\|$ (respectively, $\|\cdot\|_\infty$) means the Euclidean (respectively, the supremum) norm on $\mathbb{R}^p$, with no reference to p as the context is clear. For $W = (w_{i,j})$ a matrix of size $p_1 \times p_2$, we let $\|W\|_2 = \sup_{\|x\|=1} \|Wx\|$ be the 2-norm of W. Similarly, the ∞-norm of W is $\|W\|_\infty = \sup_{\|x\|_\infty=1} \|Wx\|_\infty = \max_{i=1,\ldots,p_1} \sum_{j=1}^{p_2} |w_{i,j}|$. We will also use the $(2,\infty)$-norm of W, i.e., $\|W\|_{2,\infty} = \sup_{\|x\|=1} \|Wx\|_\infty$. The following assumption plays a central role in our approach:

Assumption 1. *For all* $\alpha = (V_1,\ldots,V_q,c_1,\ldots,c_q) \in \Lambda$,

$$\|V_1\|_{2,\infty} \leqslant 1,\ \max(\|V_2\|_\infty,\ldots,\|V_q\|_\infty) \leqslant 1,\ \text{and } \max(\|c_i\|_\infty : i = 1,\ldots,q) \leqslant K_2,$$

where $K_2 \geqslant 0$ *is a constant.*

This type of compactness requirement has already been suggested in the statistical and machine learning community (e.g., Arjovsky et al., 2017; Anil et al., 2019; Biau et al., 2020). In the setting of this article, its usefulness is captured in the following simple but essential lemma:

Lemma 1. *Assume that Assumption 1 is satisfied. Then, for any* $k \geqslant 2$, $\mathscr{D}_k \subseteq \mathrm{Lip}_1(\mathbb{R}^d)$.

Combining Lemma 1 with Arzelà-Ascoli theorem, it is easy to see that, under Assumption 1, the class $\mathscr{D}_k$ restricted to any compact $K \subseteq \mathbb{R}^d$ is compact in the set of continuous functions on K with respect to the uniform norm. From this point of view, Assumption 1 is therefore somewhat restrictive. On the other hand, it is essential in order to guarantee that all neural networks in $\mathscr{D}_k$ are indeed 1-Lipschitz. Practically speaking, various approaches have been explored in the literature to enforce this 1-Lipschitz constraint. Gulrajani et al. (2017), Kodali et al. (2017), Wei et al. (2018), and Zhou et al. (2019) proposed a gradient penalty term, Miyato et al. (2018) applied spectral normalization, while Anil et al. (2019) have shown the empirical efficiency of the orthonormalization of Björck and Bowie (1971).

Importantly, Anil et al. (2019, Theorem 3) states that, under Assumption 1, GroupSort neural networks are universal Lipschitz approximators on compact sets. More precisely, for any Lipschitz continuous function f defined on a compact, one can find a neural network of the form (1) verifying Assumption 1 and arbitrarily close to f with respect to the uniform norm. Our objective in the present article is to explore the properties of these networks. We start in the next section by examining the case of piecewise linear functions.

3 Learning functions with a grouping size 2

For this section, we only consider GroupSort neural networks with a grouping size 2 and aim at studying their expressivity. The capacity of GroupSort networks to approximate continuous functions is studied via the representation of piecewise linear functions. For feedforward ReLU networks, their ability to represent such functions has been largely studied. In particular, Arora et al. (2018, Theorem 2.1) reveals that any piecewise linear function from $\mathbb{R}^d \to \mathbb{R}$ can be represented by a ReLU network of depth at most $\lceil \log_2(d+1) \rceil$ (the symbol $\lceil \cdot \rceil$ stands for the ceiling function), whereas He et al. (2018) specify an upper bound on their size. In the present section, we extend these results and first tackle the problem of representing piecewise linear functions with constrained GroupSort networks. Then we move to the non-linear case.

3.1 Representation of piecewise linear functions

Let us start gently by fixing the vocabulary.

Definition 1. *A continuous function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be (continuous) m_f-piecewise linear ($m_f \geqslant 2$) if there exist a partition $\Omega = \{\Omega_1, \dots, \Omega_{m_f}\}$ of $\mathbb{R}^d$ into polytopes and a collection $\ell_1, \dots, \ell_{m_f}$ of affine functions such that, for all $x \in \Omega_i$, $i = 1, \dots, m_f$, $f(x) = \ell_i(x)$.*

At this stage no further assumption is made on the sets $\Omega_1, \dots, \Omega_{m_f}$, which are just assumed to be polytopes in $\mathbb{R}^d$. An example of piecewise linear function on the real line with $m_f = 4$ is depicted in Figure 2. As this figure suggests,

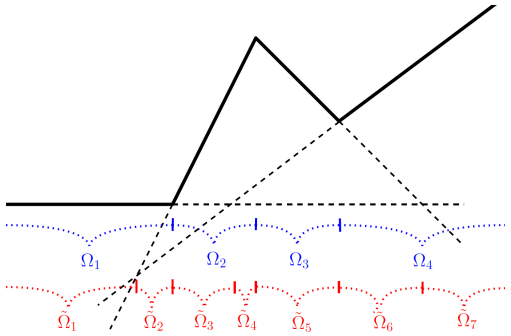


Figure 2: A 4-piecewise linear function on the real line and the associated partitions $\Omega = \{\Omega_1, \dots, \Omega_4\}$ and $\tilde{\Omega} = \{\tilde{\Omega}_1, \dots, \tilde{\Omega}_7\}$. The partition $\tilde{\Omega}$ is finer than Ω.

the ambient space $\mathbb{R}^d$ can be further covered by a second partition $\tilde{\Omega} = \{\tilde{\Omega}_1, \dots, \tilde{\Omega}_{M_f}\}$ of M_f polytopes ($M_f \geqslant 1$), in such a way that the sign of the differences $\ell_i - \ell_j$, $(i,j) \in \{1, \dots, m_f\}^2$, does not change on the subsets $\tilde{\Omega}_1, \dots, \tilde{\Omega}_{M_f}$. It is easy to see that the partition $\tilde{\Omega}$ is finer than Ω since, for each $i \in \{1, \dots, M_f\}$ there exists $j \in \{1, \dots, m_f\}$ such that $\tilde{\Omega}_i \subseteq \Omega_j$. This implies in particular that $M_f \geqslant m_f$.

The usefulness of the partition $\tilde{\Omega}$ is demonstrated by He et al. (2018, Theorem 5.1), which states that any m_f-piecewise linear function f can be written as

$$f = \max_{1 \leqslant k \leqslant M_f} \min_{i \in S_k} \ell_i, \tag{2}$$

where each S_k is a non-empty subset of $\{1, \dots, m_f\}$. This characterization of the function f is interesting, since it shows that any m_f-piecewise linear function can be computed using only a finite number of max and min operations. As identity (2) is essential for our approach, this justifies spending some time examining it.

Lemma 2. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an m_f-piecewise linear function. Then $m_f \leqslant M_f \leqslant \min(2^{m_f^2/2}, (m_f/\sqrt{2})^{2d})$.*

Lemma 2 is an improvement of He et al. (2018, Lemma 5.1), which shows that $M_f \leqslant m_f!$. Our proof method exploits the inequality $M_f \leqslant C_{m_f(m_f-1)/2,d}$, where $C_{n,d}$ denotes the number of arrangements of n hyperplanes in a space of dimension d (Devroye et al., 1996, Chapter 5). Another application of (2) is encapsulated in Proposition 1 below, which will be useful for later analysis, in combining maxima and minima in neural networks of the form (1).

Proposition 1. *Let $f_1, \dots, f_m : \mathbb{R}^d \to \mathbb{R}$ be a collection of functions ($m \geqslant 2$), each represented by a neural network of the form (1), with common depth q and sizes s_i, $i = 1, \dots, m$.*

In the specific case where $m = 2^n$ for some $n \geqslant 1$, there exist neural networks of the form (1) (with grouping size 2) with depth $q + \log_2(m)$ and size at most $s_1 + \cdots + s_m + m - 1$ that represent the functions $f = \max(f_1, \dots, f_m)$ and $g = \min(f_1, \dots, f_m)$.

If m is arbitrary, then there exist neural networks of the form (1) with depth $q + \lceil \log_2(m) \rceil$ and size at most $s_1 + \cdots + s_m + 2m - 1$ that represent the functions f and g.

Interestingly, Arora et al. (2018, Lemma D.3), which is the analog of Proposition 1 asserts that the size with ReLU activations is at most $s_1 + \cdots + s_m + 8m - 4$. For the specific computation of maxima/minima of functions, it should be stressed that GroupSort activations slightly reduces the size of the networks. By combining Lemma 2, Proposition 1, and identity (2), we are led to the following theorem, which reveals the ability of GroupSort networks for representing 1-Lipschitz piecewise linear functions.

Theorem 1. *Let $f \in \mathrm{Lip}_1(\mathbb{R}^d)$ that is also m_f-piecewise linear. Then there exists a neural network of the form (1) verifying Assumption 1 that represents f. Besides, its depth is $\lceil \log_2(M_f) \rceil + \lceil \log_2(m_f) \rceil + 1$ and its size is at most $3m_f M_f + M_f - 1$.*

This result should be compared with state-of-the-art results known for ReLU neural networks. In particular, Arora et al. (2018, Theorem 2.1) reveals that any m_f-piecewise linear function f can be represented by a ReLU network with depth at most $\lceil \log_2(d+1) \rceil$. The upper bound of Theorem 1 can be larger since it involves both M_f and m_f. On the other hand, the upper bound $O(m_f M_f)$ on the size significantly improves on He et al. (2018, Theorem 5.2), which is at least $O(d 2^{m_f M_f})$. This improvement in terms of size can be roughly explained by the depth/size trade-off results known in deep learning theory. As a matter of fact, many theoretical research papers have underlined the benefits of depth relatively to width for parameterizing complex functions (as, for example, in Telgarsky, 2015, 2016). For a fixed number of neurons, when comparing two neural networks, the deepest is the most expressive one (Lu et al., 2017).

It turns out that Theorem 1 can be significantly refined when the partition Ω satisfies some geometrical properties. Our next proposition examines the case where the sets $\Omega_1, \ldots, \Omega_{m_f}$ are convex.

Corollary 1. *Let $f \in \mathrm{Lip}_1(\mathbb{R}^d)$ that is also m_f-piecewise linear with convex subdomains $\Omega_1, \ldots, \Omega_{m_f}$. Then there exists a neural network of the form (1) verifying Assumption 1 that represents f. Besides, its depth is $2\lceil \log_2(m_f) \rceil + 1$ and its size is at most $3m_f^2 + m_f - 1$.*

Corollary 1 offers a significant improvement over Theorem 1, since in general $M_f \gg m_f$. We note in passing that the result of this proposition is dimension-free.

3.2 GroupSort neural networks on the real line

Piecewise linear functions defined on $\mathbb{R}$ deserve a special treatment, since in this case, any connected subset is convex.

Proposition 2. *Let $f \in \mathrm{Lip}_1(\mathbb{R})$ that is also m_f-piecewise linear. Then there exists a neural network of the form (1) verifying Assumption 1 that represents f. Besides, its depth is $2\lceil \log_2(m_f) \rceil + 1$ and its size is at most $3m_f^2 + m_f - 1$.*

In the specific case where f is convex (or concave), then there exists a neural network of the form (1) verifying Assumption 1 that represents f. Its depth is $\lceil \log_2(m_f) \rceil + 1$ and its size is at most $3m_f - 1$.

When f is convex (or concave) and $m_f = 2^n$ for some $n \geqslant 1$, then there exists a neural network of the form (1) verifying Assumption 1 that represents f. Its depth is $\log_2(m_f) + 1$ and its size is at most $2m_f - 1$.

This proposition is the counterpart of Arora et al. (2018, Theorem 2.2), which states that any m_f-piecewise linear function from $\mathbb{R} \to \mathbb{R}$ can be represented by a 2-layer ReLU neural network with a size at least $m_f - 1$. He et al. (2018, Theorem 5.2) shows that the upper-bound on the size of ReLU networks is $O(2^{m^2 + 2(m-1)})$. Thus, for the representation of piecewise linear functions on the real line, GroupSort networks require larger depths but smaller sizes. Besides, bear in mind that the obtained ReLU neural networks do not necessarily verify a requirement similar to the one of Assumption 1.

Regarding the number of linear regions of GroupSort networks on the real line, we have the following result:

Lemma 3. *Any neural network of the form (1) on the real line, with depth q and widths $v_1, \ldots, v_{q-1}$, parameterizes a piecewise linear function with at most $2^{q-2} \times (v_1/2 + 1) \times v_2 \times \cdots \times v_{q-1}$ linear subdomains.*

We deduce from this lemma that for a neural network of the form (1) with depth $q \geqslant 2$ and constant width v, the maximum number of linear regions is $O(2^{q-3} v^{q-1})$. Similarly to ReLU networks (Montúfar et al., 2014; Arora et al., 2018), the maximum number of linear regions for GroupSort networks with grouping size 2 is also likely to grow polynomially in v and exponentially in q.

Our next corollary now illustrates the trade-off between depth and width for GroupSort neural networks.

Corollary 2. *Let $f \in \mathrm{Lip}_1(\mathbb{R})$ be an m_f-piecewise linear function. Then, any neural network of the form (1) verifying Assumption 1 and representing f with a depth q, has a size s at least $\frac{1}{2}(q-1) m_f^{1/(q-1)}$.*

The lower bound highlighted in Corollary 2 is dependent on the depth q of the neural network. By looking at the minimum of the function, we get that any neural network representing f has a size $s \geqslant \frac{e \ln(m_f)}{2}$. Thus, merging this result with Proposition 2, we have that for any m_f-piecewise linear function from $\mathbb{R} \to \mathbb{R}$, there exists a GroupSort network verifying Assumption 1 with a size s satisfying

$$\frac{e \ln(m_f)}{2} \leqslant s \leqslant 3m_f^2 - m_f - 3.$$

We realize that this inequality is large but, up to our knowledge, this is first of this type for GroupSort neural networks.

3.3 Approximating Lipschitz continuous functions on compact sets

Following our plan, we tackle in this subsection the task of approximating Lipschitz continuous functions on compact sets using GroupSort neural networks. The space of continuous functions on $[0,1]^d$ is equipped with the uniform norm

$$\|f-g\|_\infty = \max_{x\in[0,1]^d} |f(x)-g(x)|.$$

The main result of the section, and actually of the article, is that GroupSort neural networks are well suited for approximating functions in $\mathrm{Lip}_1([0,1]^d)$.

Theorem 2. *Let $\varepsilon > 0$ and $d \geqslant 2$, $f \in \mathrm{Lip}_1([0,1]^d)$. Then there exists a neural network D of the form (1) verifying Assumption 1 such that $\|f-D\|_\infty \leqslant \varepsilon$. The depth of D is $O(d^2 \log_2(\frac{2\sqrt{d}}{\varepsilon}))$ and its size is $O((\frac{2\sqrt{d}}{\varepsilon})^{d^2})$.*

To the best of our knowledge, Theorem 2 is the first one that provides an upper bound on the depth and size of neural networks, with constraints on the weights, that approximate Lipschitz continuous functions.

As for the representation of piecewise linear functions, one can, for the sake of completeness, compare this bound with those previously found in the literature of ReLU neural networks. Yarotsky (2017) establishes the density of ReLU networks in Sobolev spaces, using a different technique of proof. In particular, Theorem 1 of this paper states that for any $f \in \mathrm{Lip}_1([0,1]^d)$ continuously differentiable, there exists a ReLU neural network approximating f with precision ε, with depth at most $c(\ln(1/\varepsilon)+1)$ and size at most $c\varepsilon^{-d}(\ln(1/\varepsilon)+1)$ (with a constant c function of d). Comparing this result with our Theorem 2, we see that, with respect to ε, both depths are similar but ReLU networks are smaller in size. However, one has to keep in mind that both lines of proof largely differ. Besides, our formulation ensures that the approximator is also a 1-Lipschitz function, a feature that cannot be guaranteed under the formulation of Yarotsky (2017).

It turns out however that our framework provides smaller neural networks as soon as $d = 1$.

Proposition 3. *Let $\varepsilon > 0$ and $f \in \mathrm{Lip}_1([0,1])$. Then there exists a neural network D of the form (1) verifying Assumption 1 such that $\|f-D\|_\infty \leqslant \varepsilon$. The depth of D is $2\lceil \log_2(1/\varepsilon)\rceil + 1$ and its size is $O((\frac{1}{\varepsilon})^2)$.*

Besides, if f is assumed to be convex or concave, then the depth of D is $\lceil \log_2(1/\varepsilon)\rceil + 1$ and its size is $O(\frac{1}{\varepsilon})$.

4 Impact of the grouping size

The previous section paved the way for a better understanding of GroupSort neural networks and their ability to approximate Lipschitz continuous functions. As mentioned in Section 2, one can play with the grouping size k of the neural network when defining its architecture. However, it is not clear how changing this parameter might influence the expressivity of the network. The present section aims at bringing some understanding. Following a similar reasoning as in Section 3, we start by analyzing how GroupSort networks with an arbitrary grouping size $k \geqslant 2$ can represent any piecewise linear functions:

Proposition 4 (Extension of Proposition 1)**.** *Let $f_1,\dots,f_m : \mathbb{R}^d \to \mathbb{R}$ be a collection of functions ($m \geqslant 2$), each represented by a neural network of the form (1), with common depth q and sizes s_i, $i = 1,\dots,m$.*

In the specific case where $m = k^n$ for some $n \geqslant 1$, there exist neural networks of the form (1) (with grouping size k) with depth $q + \log_k(m)$ and size at most $s_1 + \cdots + s_m + \frac{m-1}{k-1} - 1$ that represent the functions $f = \max(f_1,\dots,f_m)$ and $g = \min(f_1,\dots,f_m)$.

Similarly to Section 3, this leads to the following corollary:

Corollary 3 (Extension of Corollary 1)**.** *Let $f \in \mathrm{Lip}_1(\mathbb{R}^d)$ that is also m_f-piecewise linear with convex subdomains $\Omega_1,\dots,\Omega_{m_f}$ such that $m_f = k^n$ for some $n \geqslant 1$. Then there exists a neural network of the form (1) verifying Assumption 1 that represents f. Besides, its depth is $2\lceil \log_k(m_f)\rceil + 1$ and its size is at most $\frac{m_f^2-1}{k-1}$.*

Proposition 4 and Corollary 3 exhibit the nice properties of using larger grouping sizes. Indeed, for a given $q \geqslant 1$, there exists a neural network with depth $2q+1$ and grouping size k representing a function with k^q pieces. Consequently, the use of larger grouping sizes helps have more expressive neural networks. The efficiency of larger grouping sizes may also be explained by the following result for GroupSort networks on the real line:

Lemma 4 (Extension of Lemma 3)**.** *Any neural network of the form (1) on the real line, with depth q, widths $\nu_1,\dots,\nu_{q-1}$, and grouping size k, parameterizes a piecewise linear function with at most $k^{q-2} \times (\frac{(k-1)\nu_1}{2} + 1) \times \nu_2 \times \cdots \times \nu_{q-1}$ linear subdomains.*

Thus, the number of linear regions of a GroupSort network is likely to increase polynomially with the grouping size, which highlights the benefits of using larger groups. Similarly to Section 3, when moving to the approximation of Lipschitz continuous functions on $[0,1]^d$, we are lead to the following theorem:

Theorem 3 (Extension Theorem 2)**.** *Let $\varepsilon > 0$, $d \geqslant 2$, and $f \in \mathrm{Lip}_1([0,1]^d)$. Then there exists a neural network D of the form (1) verifying Assumption 1 with grouping size*

Methods	Up Depth	Up Size	Down Size	Reference
Representing $m = k^n$-PWL functions in $\mathbb{R}^d$ with a constant width ν				
ReLU	$\lceil \log_2(d+1) \rceil + 1$	$O(d2^{m^2})$	$O(m)$	He et al. (2018)
GroupSort $GS = k$	$\lceil 2\log_k(m) \rceil + 1$	$\frac{m^2-1}{k-1}$	$\frac{\nu \log_k(m)}{2\log_k(\nu)}$	present article
Approximating 1-Lipschitz continuous functions in $[0,1]^d$				
ReLU	$O(\ln(\frac{1}{\varepsilon}))$	$O(\frac{\ln(1/\varepsilon)}{\varepsilon^d})$	\	Yarotsky (2017)
GroupSort $GS = \lceil \frac{2\sqrt{d}}{\varepsilon} \rceil$	$O(d^2)$	$O((\frac{2\sqrt{d}}{\varepsilon})^{d^2-1})$	\	present article
Approximating 1-Lipschitz continuous functions in $[0,1]$				
ReLU (PWL representation)	2	$O(2^{1/\varepsilon^2+2/\varepsilon})$	\	He et al. (2018)
ReLU (different approach)	$O(\ln(\frac{1}{\varepsilon}))$	$O(\frac{\ln(1/\varepsilon)}{\varepsilon})$	\	Yarotsky (2017)
Adaptative ReLU	6	$O(\frac{1}{\varepsilon \ln(1/\varepsilon)})$	\	Yarotsky (2017)
GroupSort $GS = \lceil \frac{1}{\varepsilon} \rceil$	3	$O(\frac{1}{\varepsilon})$	\	present article

Table 1: Summary of the results shown in the present paper together with results previously found for ReLU networks. "Up Depth" refers to upper bounds on the depths, "Up Size" to upper bounds on the sizes, and "Down Size" to lower bounds on the sizes. The symbol "\" means that no result is known (up to our knowledge).

$\lceil \frac{2\sqrt{d}}{\varepsilon} \rceil$ *such that* $\|f - D\|_\infty \leqslant \varepsilon$. *The depth of* D *is* $O(d^2)$ *and its size is* $O((\frac{2\sqrt{d}}{\varepsilon})^{d^2-1})$.

Using a grouping size proportional to $1/\varepsilon$, we thus have a bound on the depth that is independent from the error rate. The uni-dimensional case leads to a different result:

Proposition 5 (Extension of Proposition 3)**.** *Let* $\varepsilon > 0$ *and* $f \in \mathrm{Lip}_1([0,1])$. *Then there exists a neural network* D *of the form* (1) *verifying Assumption 1 (with grouping size* k*) such that* $\|f - D\|_\infty \leqslant \varepsilon$. *The depth of* D *is* $2\lceil \log_k(\frac{1}{\varepsilon}) \rceil + 1$ *and its size is at most* $O(\frac{1}{k\varepsilon^2})$.

In particular, if k *is chosen to be equal to* $\lceil \frac{1}{\varepsilon} \rceil$*, then the depth of* D *is* 3 *and its size is* $O(\frac{1}{\varepsilon})$.

When approximating real-valued functions, the use of larger grouping sizes can significantly decrease the required size since it goes from $O(1/\varepsilon^2)$ in Proposition 3 to $O(1/\varepsilon)$ in Proposition 5. When f is assumed to be convex or concave, the depth of the network D can further be reduced to 2.

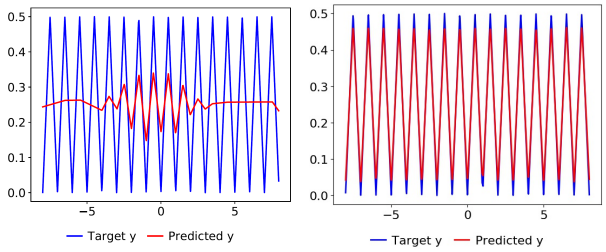
Using a different approach for approximating Lipschitz continuous functions in $[0,1]$, Yarotsky (2017, Theorem 1) shows that ReLU networks with a depth of $O(\ln(1/\varepsilon))$ is needed together with a size $O(\frac{\ln(1/\varepsilon)}{\varepsilon})$ to approximate with an error rate ε. To sum-up, when compared with ReLU networks, GroupSort neural networks with well-chosen grouping size can be significantly more expressive.

Table 1 summarizes the results shown in the present paper together with results previously found for ReLU networks. Bear in mind that GroupSort neural networks also have the supplementary condition that any parameterized function verifies the 1-Lipschitz continuity.

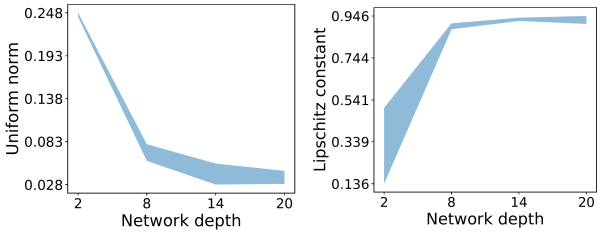
5 Experiments

Anil et al. (2019) have already compared the performances of GroupSort neural networks with their ReLU counterparts, both with constraints on the weights. In particular, they showed that ReLU neural networks are more sensitive to adversarial attacks while stressing the fact that if their weights are limited, then these networks lose their expressive power. Building on these observations, we further illustrate the good behavior of GroupSort neural networks in the context of estimating a Lipschitz continuous regression function and in approximating the Wasserstein distance (via its dual form) between pairs of distributions.

Impact of the depth. We start with the problem of learning a function f in the model $Y = f(X)$, where X follows a uniform distribution on $[-8,8]$ and f is 32-piecewise linear. To this aim, we use neural networks of the form (1) with respective depth $q = 2$, 8, 14, 20, and a constant width $\nu = 50$. Since we are only interested in the approximation properties of the networks, we assume to have at hand an infinite number of pairs $(X_i, f(X_i))$ and train the models by minimizing the mean squared error. We give in the Appendix, the full details of our experimental setting. The quality of the estimation is evaluated using the uniform norm between the target function f and the output network. In order to enforce Assumption 1, GroupSort neural networks are constrained using the orthonormalization of Björck and Bowie (1971). The results are presented in Figure 3. Note that throughout this section, confidence intervals are computed over 20 runs. In line with Theorem 1, which states that f is representable by a neural network of the form (1) with size at most $3 \times 32^2 + 32 - 1 = 3104$, we clearly observe that, as the depth of the networks increases, the uniform norm de-



(a) $q = 2$ (b) $q = 20$

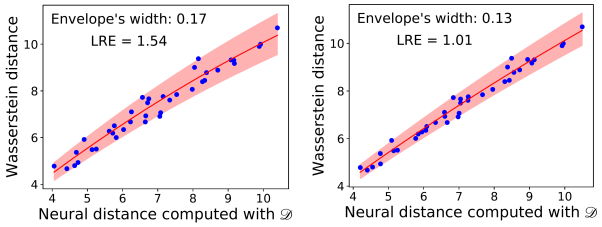


(c) Uniform norm (d) Lipschitz constants

Figure 3: Reconstruction of a 32-piecewise linear function on $[-8,8]$ with a GroupSort neural network of the form (1) with depth $q = 2$, 8, 14, 20, and a constant width $v = 50$ (the thickness of the line represents a 95%-confidence interval).

creases and the Lipschitz constant of the network converges to 1. The reconstruction of this piecewise linear function is even almost perfect for the depth $q = 20$, i.e., with a network of size only $20 \times 60 = 1200$, a value significantly smaller than the upper bound of the theorem.

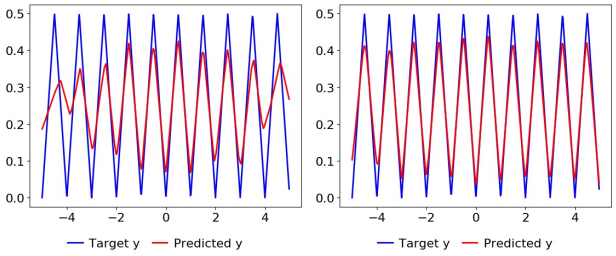
We also illustrate the behavior of GroupSort neural networks in the context of WGANs (Arjovsky et al., 2017). We run a series of small experiments in the simplified setting where we try to approximate the 1-Wasserstein distance between two bivariate mixtures of independent Gaussian distributions with 4 components. We consider networks of the form (1) with grouping size 2, a depth $q = 2$ and $q = 5$, and a constant width $v = 20$. For a pair of distributions (μ, ν), our goal is to exemplify the relationship between the 1-Wasserstein distance $\sup_{f \in \mathrm{Lip}_1(\mathbb{R}^2)}(\mathbb{E}_\mu - \mathbb{E}_\nu)$ (approximated with the Python package by Flamary and Courty, 2017) and the neural distance $\sup_{f \in \mathscr{D}_2}(\mathbb{E}_\mu - \mathbb{E}_\nu)$ (Arora et al., 2017) computed over the class of functions $\mathscr{D}_2$. To this aim, we randomly draw 40 different pairs of distributions. Then, for each of these pairs, we compute an approximation of the 1-Wasserstein distance and calculate the corresponding neural distance. Figure 4 depicts the best parabolic fit between 1-Wasserstein and neural distances, and shows the corresponding Least Relative Error (LRE) together with the width of the envelope. The take-home message of this figure is that both the LRE and the width are significantly smaller for deeper GroupSort neural networks.



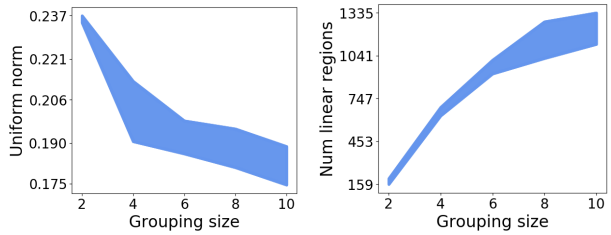
(a) $q = 2$ (b) $q = 5$

Figure 4: Scatter plots of 40 pairs of Wasserstein and neural distances computed with GroupSort neural networks, for $q = 2, 5$. The underlying distributions are bivariate Gaussians. The red curve is the optimal parabolic fitting and LRE refers to the Least Relative Error. The red zone is the envelope obtained by stretching the optimal curve.

Impact of the grouping size. To highlight the benefits of using larger grouping sizes, we show the impact of increasing the grouping size from 2 in Figure 5a to 5 in Figure 5b for the representation of a 20-piecewise linear function. This is corroborated by Figure 5c, which illustrates that the uniform norm with a 64-piecewise linear function decreases when the grouping size increases. As already underlined in Lemma 4, this may be explained by the fact that the number of linear regions significantly grows with the grouping size—see Figure 5d.

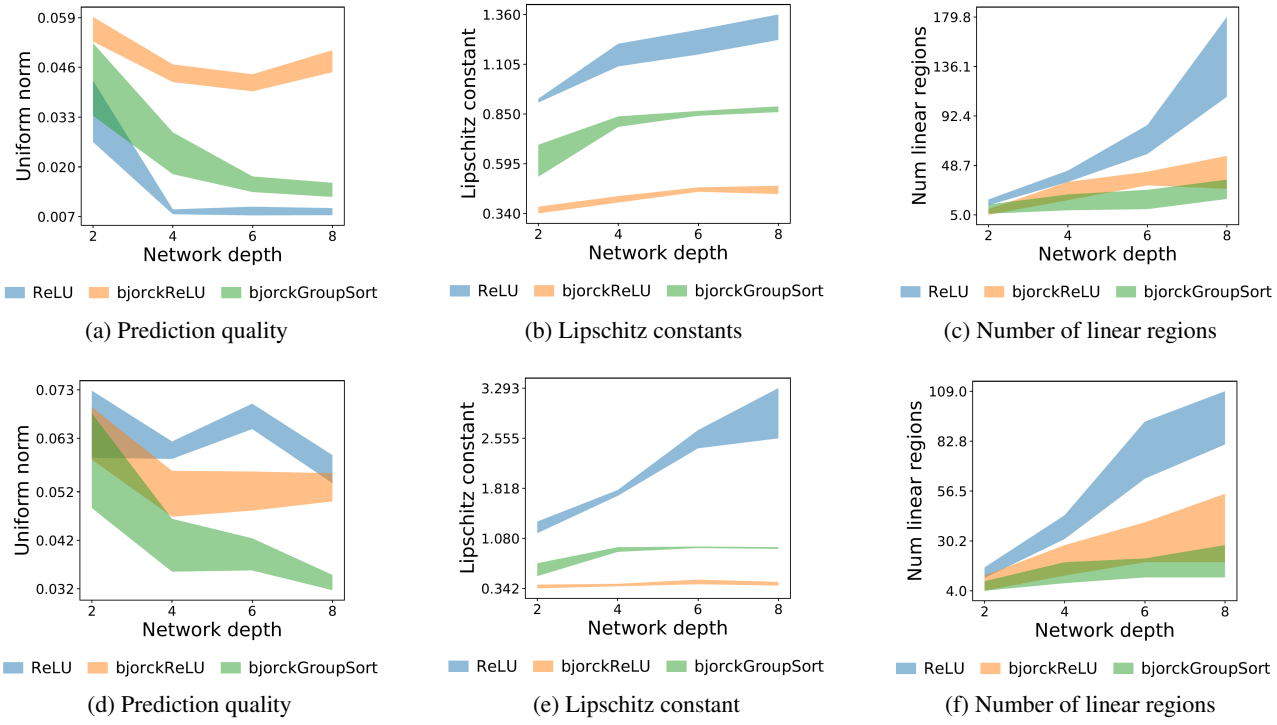


(a) Grouping size = 2 (b) Grouping size = 6



(c) Uniform norm (d) Linear regions

Figure 5: Reconstruction of a 20-piecewise linear function on $[-5,5]$ (top line) and a 64-piecewise linear function (bottom line) with GroupSort neural networks of the form (1) with depth $q = 4$ and varying grouping sizes $k = 2, 4, 6, 8, 10$.



(a) Prediction quality (b) Lipschitz constants (c) Number of linear regions

(d) Prediction quality (e) Lipschitz constant (f) Number of linear regions

Figure 6: (Top line) Estimating the function $f(x) = (1/15)\sin(15x)$ on $[0,1]$ in the model $Y = f(X)$, with a dataset of size $n = 100$. (Bottom line) Estimating the function $f(x) = (1/15)\sin(15x)$ on $[0,1]$ in the model $Y = f(X) + \varepsilon$, with a dataset of size $n = 100$ (the thickness of the line represents a 95%-confidence interval).

Comparison with ReLU neural networks. Next, in a second series of experiments, we compare the performances of GroupSort networks against two baselines: ReLU neural networks without constraints on the weights (dense in the set of continuous functions on a compact set; see Yarotsky, 2017), and ReLU neural networks with orthonormalization of Björck and Bowie (1971). The architecture of the ReLU neural networks in terms of depth and width is the same as for GroupSort networks: $q = 2, 4, 6, 8$, and $w = 20$. The task is now to approximate the 1-Lipschitz continuous function $f(x) = (1/15)\sin(15x)$ on $[0,1]$ in the models $Y = f(X)$ (noiseless case) and $Y = f(X) + \varepsilon$ (noisy case), where X is uniformly distributed on $[0,1]$ and ε follows a Gaussian distribution with standard deviation 0.05. In both cases, we assume to have at hand a finite sample of size $n = 100$ and fit the models by minimizing the mean squared error.

Both results (noiseless case and noisy case) are presented in Figure 6. We observe that in the noiseless setting Figure 6a, 6b, and 6c, ReLU neural networks without normalization have a slightly better performance with respect to the uniform norm with, however, a Lipschitz constant larger than 1. On the other hand, in the noisy case, ReLU neural networks without constraints have a tendency to overfitting (a high Lipschitz constant close to 2.7), leading to a deteriorated performance, contrary to GroupSort neural networks. Furthermore, in both cases (noiseless and noisy), ReLU with constraints are found to perform worse (due to a Lipschitz constant much smaller than 1) than their GroupSort counterparts in terms of prediction. Interestingly, we see in the two examples shown in Figure 6e and Figure 6f, that the number of linear regions for GroupSort neural networks is smaller than for ReLU networks.

Finally, we quickly show in Appendix a comparison between GroupSort and ReLU networks when approximating Wasserstein distances. The take home message is that, on this specific task, GroupSort networks perform better.

6 Conclusion

The results presented in this article show the advantage of using GroupSort neural networks over standard ReLU networks. On the one hand, ReLU neural networks without any constraints are sensitive to adversarial attacks (as they may have a large Lipschitz constant) and, on the other hand, lose expressive power when enforcing limits on their weights. On the opposite, GroupSort neural networks with constrained weights are proved to be both robust and expressive, and are therefore an interesting alternative. Moreover, by allowing larger grouping sizes for GroupSort networks, one can further increase their expressivity. These properties open new perspectives for broader use of GroupSort networks.

References

Anil, C., Lucas, J., and Grosse, R. (2019). Sorting out Lipschitz function approximation. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 291–301. PMLR.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In Precup, D. and Teh, Y., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 214–223. PMLR.

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. (2018). Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*.

Arora, S., Ge, R., Liang, Y., Ma, T., and Zhang, Y. (2017). Generalization and equilibrium in generative adversarial nets (GANs). In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 224–232.

Biau, G., Cadre, B., Sangnier, M., and Tanielian, U. (2020). Some theoretical properties of GANs. *The Annals of Statistics*, 48:1539–1566.

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

Blanchet, J., Kang, Y., and Murthy, K. (2019). Robust Wasserstein profile inference and applications to machine learning. *Journal of Applied Probability*, 56:830–857.

Cooper, D. (1995). Learning Lipschitz functions. *International Journal of Computer Mathematics*, 59:15–26.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314.

Devroye, L., Györfi, L., and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. Springer, New York.

Esfahani, P. and Kuhn, D. (2018). Data-driven distributionally robust optimization using the Wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming*, 171:115–166.

Flamary, R. and Courty, N. (2017). POT: Python Optimal Transport library.

Gao, R., Chen, X., and Kleywegt, A. (2017). Wasserstein distributional robustness and regularization in statistical learning. *arXiv:1712.06050*.

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

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of Wasserstein GANs. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5767–5777. Curran Associates, Inc.

He, J., Li, L., Xu, J., and Zheng, C. (2018). ReLU deep neural networks and linear finite elements. *arXiv:1807.03973*.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366.

Huster, T., Chiang, C.-Y. J., and Chadha, R. (2018). Limitations of the Lipschitz constant as a defense against adversarial examples. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 16–29. Springer.

Kodali, N., Abernethy, J., Hays, J., and Kira, Z. (2017). On convergence and stability of GANs. *arXiv.1705.07215*.

Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. (2017). The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems*, pages 6231–6239.

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

Montúfar, G., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 27*, pages 2924–2932. Curran Associates, Inc.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Dickstein, J. (2017). On the expressive power of deep neural networks. In Precup, D. and Teh, Y., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2847–2854. PMLR.

Seidel, R. (1995). The upper bound theorem for polytopes: An easy proof of its asymptotic version. *Computational Geometry*, 5:115–116.

Telgarsky, M. (2015). Representation benefits of deep feedforward networks. *arXiv 1509.08101*.

Telgarsky, M. (2016). Benefits of depth in neural networks. In Feldman, V., Rakhlin, A., and Shamir, O., editors, *29th Annual Conference on Learning Theory*, volume 49, pages 1517–1539. PMLR.

Villani, C. (2008). *Optimal Transport: Old and New*. Springer, Berlin.

Wei, X., Gong, B., Liu, Z., Lu, W., and Wang, L. (2018). Improving the improved training of Wasserstein GANs: A consistency term and its dual effect. *arXiv:1803.01541*.

Yarotsky, D. (2017). Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114.

Zhou, Z., Liang, J., Song, Y., Yu, L., Wang, H., Zhang, W., Yu, Y., and Zhang, Z. (2019). Lipschitz generative adversarial nets. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 7584–7593. PMLR.

A Technical results and complementary experiments

A.1 Proof of Lemma 1

We prove the result for $\mathscr{D}_2$. The result for $\mathscr{D}_k$ holds following a similar argument.

Fix $D_{2,\alpha} \in \mathscr{D}_2$, $\alpha \in \Lambda$. According to (1), we have, for $x \in \mathbb{R}^d$, $D_{2,\alpha}(x) = f_q \circ \cdots \circ f_1(x)$, where $f_i(t) = \sigma_2(V_i t + c_i)$ for $i = 1, \ldots, q-1$ (σ_2 is applied on pairs of components), and $f_q(t) = V_q t + c_q$. Therefore, for $(x,y) \in (\mathbb{R}^d)^2$,

$$\begin{aligned}
\|f_1(x) - f_1(y)\|_\infty &\leqslant \|V_1 x - V_1 y\|_\infty \\
&\quad \text{(since } \sigma_2 \text{ is 1-Lipschitz)} \\
&= \|V_1(x-y)\|_\infty \\
&\leqslant \|V_1\|_{2,\infty} \, \|x-y\| \\
&\leqslant \|x-y\| \\
&\quad \text{(by Assumption 1).}
\end{aligned}$$

Thus,

$$\begin{aligned}
\|f_2 \circ f_1(x) - f_2 \circ f_1(y)\|_\infty &\leqslant \|V_2 f_1(x) - V_2 f_1(y)\|_\infty \\
&\quad \text{(since } \sigma_2 \text{ is 1-Lipschitz)} \\
&\leqslant \|V_2\|_\infty \, \|f_1(x) - f_1(y)\|_\infty \\
&\leqslant \|f_1(x) - f_1(y)\|_\infty \\
&\quad \text{(by Assumption 1)} \\
&\leqslant \|x-y\|.
\end{aligned}$$

Repeating this, we conclude that, for each $\alpha \in \Lambda$ and all $(x,y) \in (\mathbb{R}^d)^2$, $|D_{2,\alpha}(x) - D_{2,\alpha}(y)| \leqslant \|x-y\|$, which is the desired result.

A.2 Proof of Lemma 2

Recall that $m_f \geqslant 2$. Throughout the proof, we let $\cdot$ refer to the dot product in $\mathbb{R}^d$. Let $(i,j) \in \{1, \ldots, m_f\}^2$, $i \neq j$. There exist $(a_i, b_i) \in \mathbb{R}^d \times \mathbb{R}$ and $(a_j, b_j) \in \mathbb{R}^d \times \mathbb{R}$ such that $\ell_i = a_i \cdot x + b_i$ and $\ell_j = a_j \cdot x + b_j$. Therefore,

$$\ell_i(x) - \ell_j(x) \leqslant 0 \iff x \cdot (a_i - a_j) \leqslant b_j - b_i.$$

So, there exist two subdomains $\tilde{\Omega}_1$ and $\tilde{\Omega}_2$, separated by an affine hyperplane, in which $\ell_i - \ell_j$ does not change sign. By repeating this operation for the $m_f(m_f-1)/2$ different pairs (ℓ_i, ℓ_j), we get that the number M_f of subdomains on which any pair $\ell_i - \ell_j$ does not change sign is smaller than the maximal number of arrangements of $m_f(m_f-1)/2$ hyperplanes.

Denoting by $C_{n,d}$ the maximal number of arrangements of n hyperplanes in $\mathbb{R}^d$, we know that when $d > n$ then $C_{n,d} = 2^n$, whereas if $n > d$ the upper bound $C_{n,d} \leqslant (1+n)^d$ becomes preferable (Devroye et al., 1996, Chapter 30). Thus, we have

$$m_f \leqslant M_f \leqslant \min\left(2^{m_f^2/2}, (m_f/\sqrt{2})^{2d}\right).$$

A.3 Proof of Proposition 1

We prove the first part of the proposition by using an induction on n. The case where $n = 1$ and thus $m = 2^1$ is clear since the function $f = \max(f_1, f_2)$ can be represented by a neural network of the form (1) with depth $q+1$ and size $s_1 + s_2 + 1$. Now, let $m = 2^n$ with $n > 1$. We have that $m/2 = 2^{n-1}$. By the induction hypothesis, $g_1 = \max(f_1, \ldots, f_{m/2})$ and $g_2 = \max(f_{m/2+1}, \ldots, f_m)$ can be represented by neural networks of the form (1) with depths $q+n-1$, and sizes at most $s_1 + \cdots + s_{m/2} + m/2 - 1$ and $s_{m/2+1} + \cdots + s_m + m/2 - 1$, respectively. Consequently, the function $G(x) = (g_1(x), g_2(x))$ can be implemented by a neural network of the form (1) with depth $q+n-1$ and size $s_1 + \cdots + s_m + m - 2$. Finally, by concatenating a one neuron layer, we have that the function $f = \max(g_1, g_2)$ can be represented by a neural network of the form (1) with depth $q+n = q+\log_2(m)$ and size at most $s_1 + \cdots + s_m + m - 1$.

Now, let us prove the case where m is arbitrary. Let $f_1, \ldots, f_m : \mathbb{R}^d \to \mathbb{R}$ be a collection of functions ($m \geqslant 2$), each represented by a neural network of the form (1) with depth q and size s_i, $i = 1, \ldots, m$. We prove below by an induction on n

that there exists a neural network of the form (1) with depth $q+\lceil\log_2(m)\rceil$, a final layer of width $v_{q-1}=2$, and a size at most $s_1+\cdots+s_m+2^{\lceil\log_2(m)\rceil}-1$ that represents the functions $f=\max(f_1,\ldots,f_m)$ and $g=\min(f_1,\ldots,f_m)$ (the symbol $\lceil\cdot\rceil$ stands for the ceiling function and the symbol $\lfloor\cdot\rfloor$ stands for the integer function).

The base case $m=2$ is clear using the GroupSort activation and $v_1=2$. For $m>2$, let $n\geqslant 2$ be such that $2^{n-1}\leqslant m<2^n$. Let $g_1=\max(f_1,\ldots,f_{2^{n-1}})$ and $g_2=\max(f_{2^{n-1}+1},\ldots,f_m)$. From the first part of the proof, we know that g_1 can be represented by a neural network of the form (1) with depth $q_1=q+\lfloor\log_2 m\rfloor=q+n-1$ and size $s_1+\cdots+s_{2^{n-1}}+2^{n-1}-1$. Also, by the induction hypothesis, g_2 can be represented by a neural network of the form (1) with depth $q_2=q+\lceil\log_2(m-2^{n-1})\rceil$ and size at most $s_{2^{n-1}+1}+\cdots+s_m+2^{\lceil\log_2(m-2^{n-1})\rceil}-1$. Therefore, by padding identity matrices with two neurons (recall that $v_{q_2-1}=2$) on layers from $q+\lceil\log_2(m-2^{n-1})\rceil$ to $q+n-1$, we have:

$$\begin{aligned}2^{\lceil\log_2(m-2^{n-1})\rceil}-1+2(n-2-\lceil\log_2(m-2^{n-1})\rceil)&=\sum_{k=0}^{k=\lceil\log_2(m-2^{n-1})\rceil-1}2^k+\sum_{k=\lceil\log_2(m-2^{n-1})\rceil}^{k=n-2}2^1\\&\leqslant\sum_{k=0}^{k=n-2}2^k=2^{n-1}-1.\end{aligned}$$

Thus, g_2 can be represented by a neural network of the form (1) with depth $q_2=q+\lfloor\log_2 m\rfloor$ and size at most $s_{2^{n-1}+1}+\cdots+s_m+2^{n-1}-1$. Now, the bivariate function $G(x)=(g_1(x),g_2(x))$ can be implemented by a neural network of the form (1) with depth $q+\lfloor\log_2(m)\rfloor$ and size s such that

$$s\leqslant s_1+\cdots+s_m+2(2^{n-1}-1)=s_1+\cdots+s_m+2^n-2.$$

By concatenating a one neuron layer, we have that the function $f=\max(g_1,g_2)$ can be represented by a neural network of the form (1) with depth $q+\lceil\log_2(m)\rceil$ and size at most $s_1+\cdots+s_m+2^n-1=s_1+\cdots+s_m+2^{\lceil\log_2 m\rceil}-1$. The conclusion follows using the inequality $2^{\lceil\log_2 m\rceil}\leqslant 2m$.

A.4 Proof of Theorem 1

Let $f\in\mathrm{Lip}_1(\mathbb{R}^d)$ that is also m_f-piecewise linear. We know that each linear function can be represented by a 1-neuron neural network verifying Assumption 1 (no need for hidden layers). It is easy to see, using a small variant of Proposition 1, that any collection of $\tilde{m}$ linear functions with $\tilde{m}\leqslant m$ can be represented by a neural network of depth $\lceil\log_2(m)\rceil+1$ and size at most $3m-1$. Thus, combining (2) with Proposition 1, for each $k\in\{1,\ldots,M_f\}$ there exists a neural network of the form (1), verifying Assumption 1 and representing the function $\min_{i\in S_k}\ell_i$, with depth equal to $\lceil\log_2(m_f)\rceil+1$ (since $|S_k|\leqslant m_f$) and size at most $3m_f-1$.

Using again Proposition 1, we conclude that there exists a neural network of the form (1), verifying Assumption 1 and representing f, with depth$\lceil\log_2(M_f)\rceil+\lceil\log_2(m_f)\rceil+1$ and size at most $3m_fM_f+M_f-1$.

A.5 Proof of Corollary 1

According to He et al. (2018, Theorem A.1), the function f can be written as

$$f=\max_{1\leqslant k\leqslant m_f}\min_{i\in S_k}\ell_i,$$

where $|S_k|\leqslant m_f$. Using the same technique of proof as for Theorem 1, we find that there exists a neural network of the form (1), verifying Assumption 1 and representing f, with depth equal to $2\lceil\log_2(m_f)\rceil+1$ and size at most $3m_f^2+m_f-1$.

A.6 Proof of Proposition 2

Let $f\in\mathrm{Lip}_1(\mathbb{R})$ that is also m_f-piecewise linear. The proof of the first statement is an immediate consequence of Corollary 1 since connected subsets of $\mathbb{R}$ are also convex.

As for the second claim of the proposition, considering the case where f is convex, we know from He et al. (2018, Theorem A.1) that f can be written as

$$f=\max_{1\leqslant k\leqslant m_f}\ell_k.$$

Each function ℓ_k, $k = 1, \dots, m_f$, can be represented by a 1-neuron neural network verifying Assumption 1. Hence, by Proposition 1, there exists a neural network of the form (1), verifying Assumption 1 and representing f, with depth $\lceil \log_2(m_f) \rceil + 1$ and size at most $3m_f - 1$.

The last claim of the proposition for $m = 2^n$ is clear using Proposition 1.

A.7 Proof of Lemma 3

The result is proved by induction on q. To begin with, in the case $q = 2$ we have a neural network with one hidden layer. When applying the GroupSort function with a grouping size 2, every activation node is defined as the max or min between two different linear functions. The maximum number of breakpoints is equal to the maximum number of intersections, that is $\nu_1/2$. Thus, there is at most $\nu_1/2 + 1$ pieces.

Now, let us assume that the property is true for a given $q \geqslant 3$. Consider a neural network with depth q and widths $\nu_1, \dots, \nu_{q-1}$. Observe that the input to any node in the last layer is the output of a $\mathbb{R} \to \mathbb{R}$ GroupSort neural network with depth $(q-1)$ and widths $\nu_1, \dots, \nu_{q-2}$. Using the induction hypothesis, the input to this node is a function from $\mathbb{R} \to \mathbb{R}$ with at most $2^{q-3} \times (\nu_1/2+1) \times \cdots \times \nu_{q-2}$ pieces. Thus, after applying the GroupSort function with a grouping size 2, each node output is a function with at most $2 \times (2^{q-3} \times (\nu_1/2+1) \times \nu_2 \times \cdots \times \nu_{q-2})$. With the final layer, we take an affine combination of ν_{q-1} functions, each with at most $2^{q-2} \times (\nu_1/2+1) \times \nu_2 \times \cdots \times \nu_{q-2}$ pieces. In all, we therefore get at most $2^{q-2} \times (\nu_1/2+1) \times \nu_2 \times \cdots \times \nu_{q-1}$ pieces. The induction step is completed.

A.8 Proof of Corollary 2

Let f be an m_f-piecewise linear function. For a neural network of depth q and widths $\nu_1, \dots, \nu_q$ representing f, we have, by Lemma 3,

$$2^{q-1} \times (\nu_1/2+1) \times \cdots \times \nu_{q-1} \geqslant m_f.$$

By the inequality of arithmetic and geometric means, minimizing the size $s = \nu_1/2 + \cdots + \nu_k$ subject to this constraint, means setting $\nu_1/2 + 1 = \nu_2 = \cdots = \nu_k$. This implies that $s \geqslant \frac{1}{2}(q-1) m_f^{1/(q-1)}$.

A.9 Proof of Theorem 2

The proof follows the one from Cooper (1995, Theorem 3). Tesselate $[0,1]^d$ by cubes of side $s = \varepsilon/(2\sqrt{d})$ and denote by $n = (\lceil 1/s \rceil)^d$ the number of cubes in the tesselation. Choose n data points, one in each different cube. Then any Delaunay sphere will have a radius $R < \varepsilon/2M_f$. Now, construct $\tilde{f}$ by linearly interpolating between values of f over the Delaunay simplices. According to Seidel (1995), the number m_f of subdomains is $O(n^{d/2})$ and each of them is convex. Besides, by Cooper (1995, Lemma 2), $\tilde{f}$ guarantees an approximation error $\|f - \tilde{f}\|_\infty \leqslant \varepsilon$.

Using Corollary 1, we know that there exists a neural network of the form (1) verifying Assumption 1 and representing $\tilde{f}$. Besides, its depth is $2\lceil \log_2(m_f) \rceil + 1$ and its size is at most $3m_f^2 + m_f - 1$. Consequently, we have that the depth of the neural network is $2\lceil \log_2(m_f) \rceil + 1 = O(d^2 \log_2(\frac{2\sqrt{d}}{\varepsilon}))$ and the size at most $O(m^2) = O((\frac{2\sqrt{d}}{\varepsilon})^{d^2})$.

A.10 Proof of Proposition 3

Let $f \in \mathrm{Lip}_1([0,1])$ and f_m be the piecewise linear interpolation of f with the following $2^m + 1$ breakpoints: $k/2^m$, $k = 0, \dots, 2^m$. We know that the function f_m approximates f with an error $\varepsilon_m \leqslant 2^{-m}$. In particular, for any $m \geqslant \log_2(1/\varepsilon)$, we have $\varepsilon_m \leqslant \varepsilon$. Besides, for any m, f_m is a 1-Lipschitz function defined on $[0,1]$, piecewise linear on 2^m subdomains. Thus, according to Proposition 2, there exists a neural network of the form (1), verifying Assumption 1 and representing f_m, with depth $2m+1$ and size at most $3 \times 2^{2m} + 2^m - 1$. Taking $m = \lceil \log_2(1/\varepsilon) \rceil$ shows the desired result.

Let $\varepsilon > 0$, let f be a convex (or concave) function in $\mathrm{Lip}_1([0,1])$, and let f_m be the piecewise linear interpolation of f with the following $2^m + 1$ breakpoints: $k/2^m$, $k = 0, \dots, 2^m$. The function f_m approximates f with an error $\varepsilon_m = 2^{-m}$. In particular, for any $m \geqslant \log_2(1/\varepsilon)$, we have $\varepsilon_m \leqslant \varepsilon$. Besides, for any m, f_m is a 2^m-piecewise linear convex function defined on $[0,1]$. Hence, by Proposition 2, there exists a neural network of the form (1), verifying Assumption 1 and representing f_m, with depth $m+1$ and size at most $2 \times 2^m - 1$. Taking $m = \lceil \log_2(1/\varepsilon) \rceil$ leads to the desired result.

A.11 Proof of Proposition 4

We prove the result by using an induction on n. The case where $n = 1$ and thus $m = k^1$ is true since the function $f = \max(f_1, \dots, f_k)$ can be represented by a neural network of the form (1) with grouping size k, depth $q+1$, and size $s_1 + \cdots + s_k + 1$. Now, let $m = k^n$ with $n > 1$. We have that $\lfloor m/k \rfloor = \lceil m/k \rceil = m/k = k^{n-1}$. Let $g_1 = \max(f_1, \dots, f_{m/k}), g_2 = \max(f_{m/k+1}, \dots, f_{2m/k}), \dots, g_k = \max(f_{((k-1)m/k)+1}, \dots, f_m)$. By the induction hypothesis, $g_1, \dots, g_k$ can all be represented by neural networks of the form (1) with grouping size k, width depths equal to $q+n-1$ and sizes at most $s_1 + \cdots + s_{m/k} + \frac{k^{n-1}-1}{k-1}, \dots, s_{(k-1)m/k+1} + \cdots + s_m + \frac{k^{n-1}-1}{k-1}$, respectively.

Consequently, the function $G(x) = (g_1(x), \dots, g_k(x))$ can be implemented by a neural network of the form (1) with grouping size k, depth $q+n-1$, and size at most $s_1 + \cdots + s_m + m - 2$. Finally, by concatenating a one neuron layer, we see that the function $f = \max(g_1, \dots, g_k)$ can be represented by a neural network of the form (1) with depth $q+n = q+\log_k(m)$ and size at most

$$s_1 + \cdots + s_m + k\Big(\frac{k^{n-1}-1}{k-1}\Big) + 1 = s_1 + \cdots + s_m + \frac{k^n - 1}{k-1} = s_1 + \cdots + s_m + \frac{m-1}{k-1}.$$

A.12 Proof of Corollary 3

According to He et al. (2018, Theorem A.1), the function f can be written as

$$f = \max_{1 \leqslant k \leqslant m_f} \min_{i \in S_k} \ell_i,$$

where $|S_k| \leqslant m_f$ and $m_f = k^n$ for some $n \geqslant 1$. It is easy to see, using a small variant of Proposition 4, that any collection of $\tilde{m}$ linear functions with $\tilde{m} \leqslant m_f$ can be represented by a neural network of depth $\log_k(m) + 1$ and size at most $\frac{m_f - 1}{k-1}$. Therefore, by Proposition 4, there exists a neural network verifying Assumption 1 with grouping size k representing $\min_{i \in S_k} \ell_i$ with depth $\log_k(m) + 1$ and size at most $\frac{m_f - 1}{k-1}$.

Using again Proposition 4, we find that there exists a neural network, verifying Assumption 1, with grouping size k, representing f with depth $2\log_k(m_f) + 1$ and size at most

$$m_f\Big(\frac{m_f - 1}{k-1}\Big) + \frac{m_f - 1}{k-1} = \frac{m_f^2 - 1}{k-1}.$$

A.13 Proof of Lemma 4

The result is proved by induction on q. To begin with, in the case $q = 2$ we have a neural network with one hidden layer. When applying the GroupSort function with a grouping size k, the maximum number of breakpoints is equal to the maximum number of intersections of linear functions. In each group of k functions, there are at most $\frac{k(k-1)}{2}$ intersections. Thus, there are at most $\frac{k(k-1)}{2} \times \frac{v_1}{k} = \frac{(k-1)v_1}{2}$ breakpoints, that is $\frac{(k-1)v_1}{2} + 1$ pieces.

Now, let us assume that the property is true for a given $q \geqslant 3$. Consider a neural network with depth q and widths $v_1, \dots, v_{q-1}$. Observe that the input to any node in the last layer is the output of a $\mathbb{R} \to \mathbb{R}$ GroupSort neural network with depth $(q-1)$ and widths $v_1, \dots, v_{q-2}$. Using the induction hypothesis, the input to this node is a function from $\mathbb{R} \to \mathbb{R}$ with at most $k^{q-3} \times (\frac{(k-1)v_1}{2} + 1) \times \cdots \times v_{q-2}$ pieces. Thus, after applying the GroupSort function with a grouping size k, each node output is a function with at most $k \times (k^{q-3} \times (\frac{(k-1)v_1}{2} + 1) \times v_2 \times \cdots \times v_{q-2})$. With the final layer, we take an affine combination of v_{q-1} functions, each with at most $k^{q-2} \times (\frac{(k-1)v_1}{2} + 1) \times v_2 \times \cdots \times v_{q-2}$ pieces. In all, we therefore get at most $k^{q-2} \times (\frac{(k-1)v_1}{2} + 1) \times v_2 \times \cdots \times v_{q-1}$ pieces. The induction step is completed.

A.14 Proof of Theorem 3

The proof of Theorem 3 is straightforward and follows the one of Theorem 2 combined with the result obtained in Corollary 3.

A.15 Proof of Proposition 5

Let $f \in \mathrm{Lip}_1([0,1])$ and f_m be the piecewise linear interpolation of f with the following k^n+1 breakpoints: $i/k^n, k=0,\dots,k^n$. We know that the function f_m approximates f with an error $\varepsilon_m \leqslant k^{-n}$. In particular, for any $n \geqslant \log_k(1/\varepsilon)$, we have $\varepsilon_n \leqslant \varepsilon$. Besides, for any n, f_{k^n} is a 1-Lipschitz function defined on $[0,1]$, piecewise linear on k^n subdomains. Thus, according to Corollary 3, there exists a neural network of the form (1), verifying Assumption 1 and representing f_{k^n}, with grouping size k, depth $2n+1$, and size at most $\frac{k^{2n}-1}{k-1}$. Taking $n = \lceil \log_k(1/\varepsilon) \rceil$ shows the desired result.

B Experiments: Extended comparison between GroupSort and ReLU networks

We provide in this section further results and details on the experiments ran in Section 5.

B.1 Task 1: Approximating functions

Piecewise linear functions. We complete the experiments of Section 5 by estimating the 6-piecewise linear function f in the model $Y = f(X)$ (noiseless case, see Figure 7 and Figure 8) and in the model $Y = f(X) + \varepsilon$ (noisy case, see Figure 9 and Figure 10). Recall that in both cases, X follows a uniform distribution on $[-1.5, 1.5]$ and the sample size is $n = 100$.

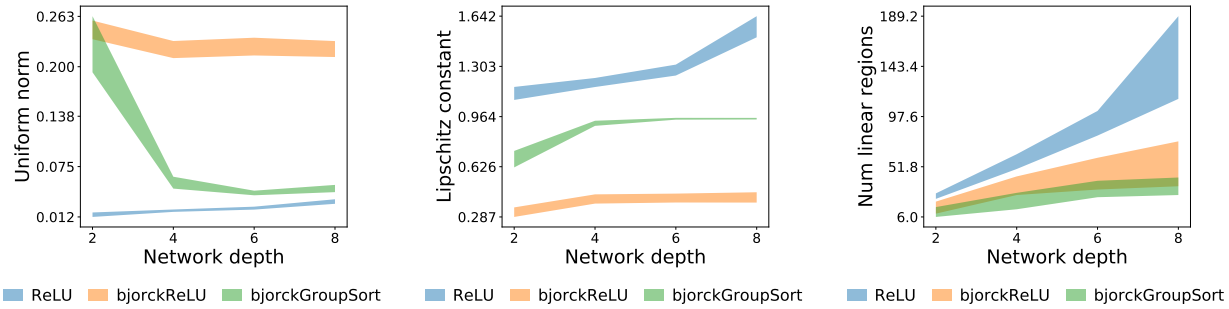
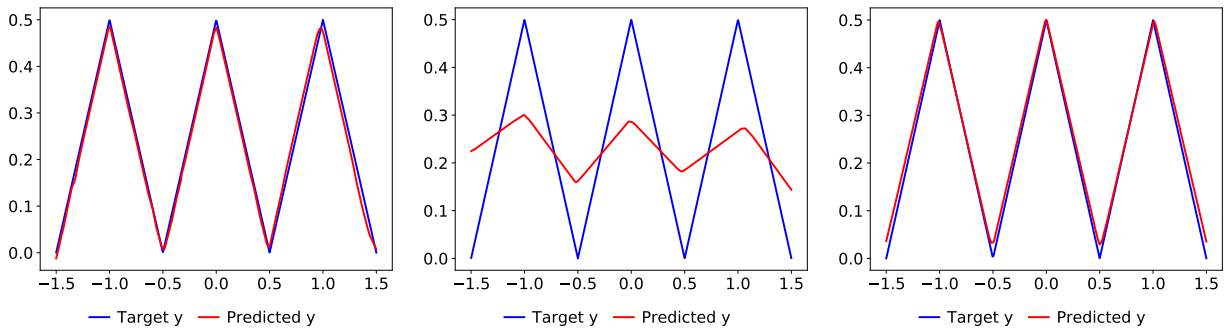


Figure 7: Estimating the 6-piecewise linear function in the model $Y = f(X)$, with a dataset of size $n = 100$ (the thickness of the line represents a 95%-confidence interval).



(a) ReLU (b) bjorckReLU (c) bjorckGroupSort

Figure 8: Reconstructing the 6-piecewise linear function in the model $Y = f(X)$, with a dataset of size $n = 100$.

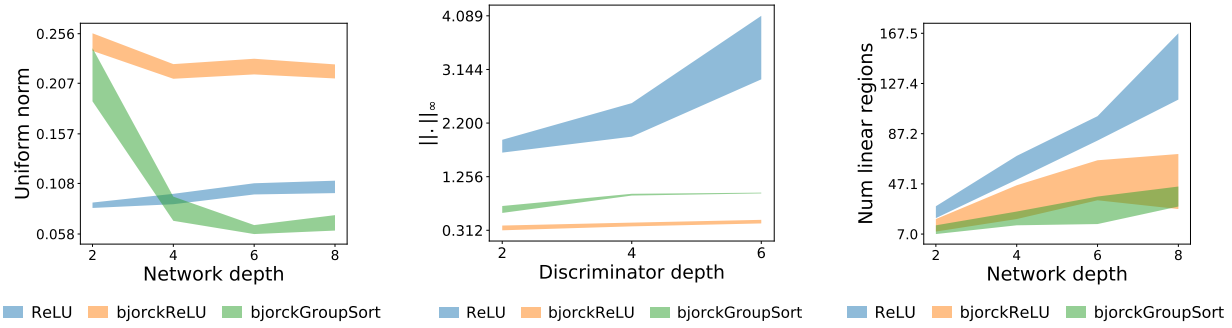
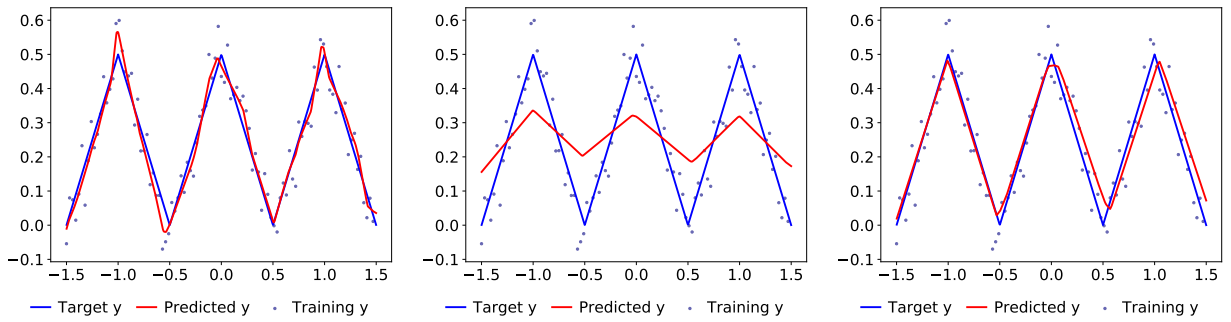


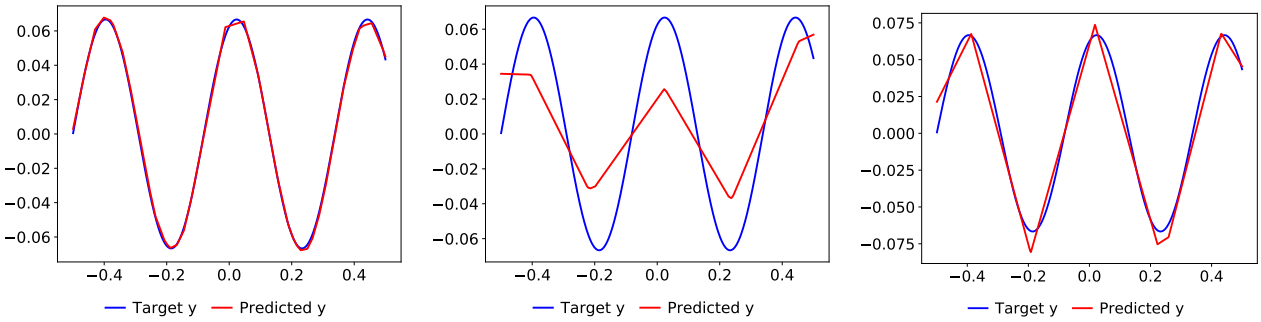
Figure 9: Estimating the 6-piecewise linear function in the model $Y = f(X) + \varepsilon$, with a dataset of size $n = 100$ (the thickness of the line represents a 95%-confidence interval).



(a) ReLU (b) bjorckReLU (c) bjorckGroupSort

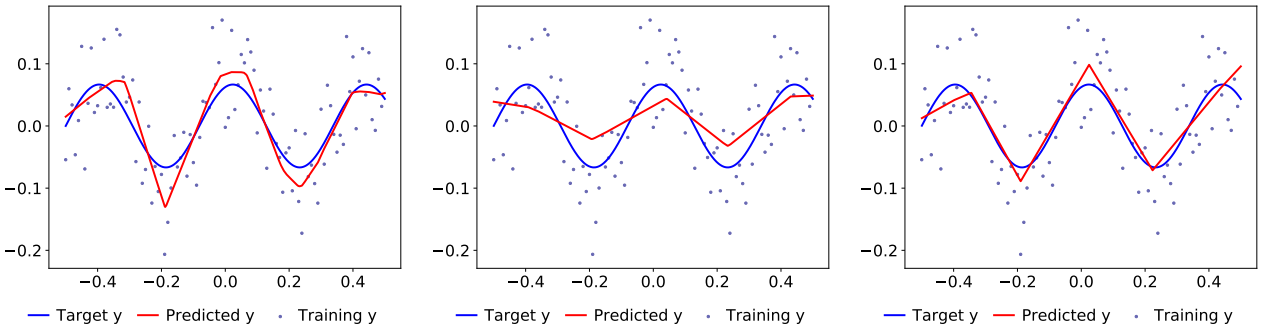
Figure 10: Reconstructing the 6-piecewise linear function in the model $Y = f(X) + \varepsilon$, with a dataset of size $n = 100$.

The sinus function. We provide in this subsection additional details for the learning of the sinus function $f(x) = (1/15)\sin(15x)$ defined on $[0, 1]$ (see Section 5). Figure 11 is the case without noise while Figure 12 is the case with noise.



(a) ReLU (b) bjorckReLU (c) bjorckGroupSort

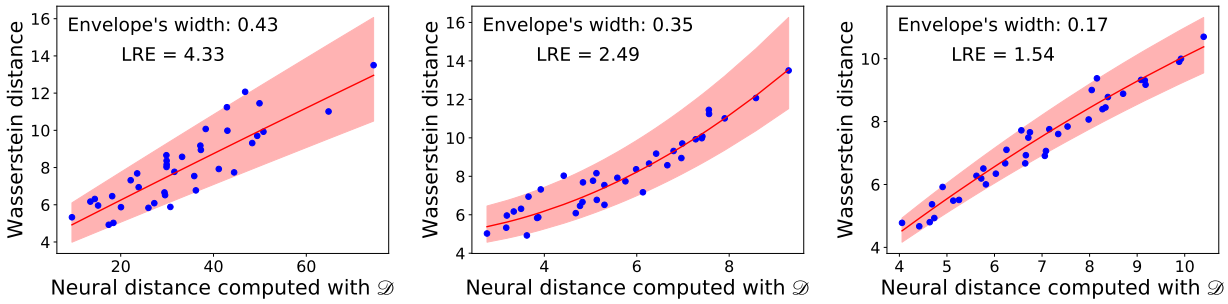
Figure 11: Reconstructing the function $f(x) = (1/15)\sin(15x)$ in the model $Y = f(X)$, with a dataset of size $n = 100$.



(a) ReLU (b) bjorckReLU (c) bjorckGroupSort

Figure 12: Reconstructing the function $f(x) = (1/15)\sin(15x)$ in the model $Y = f(X) + \varepsilon$, with a dataset of size $n = 100$.

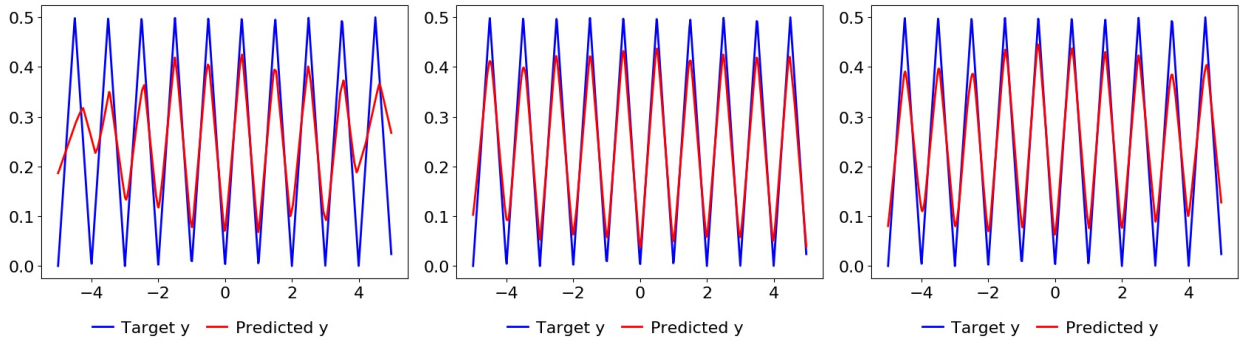
B.2 Task 2: Calculating Wasserstein distances



(a) $\mathscr{D}$ = ReLU network (b) $\mathscr{D}$ = bjorckReLU network (c) $\mathscr{D}$ = bjorckGroupSort network

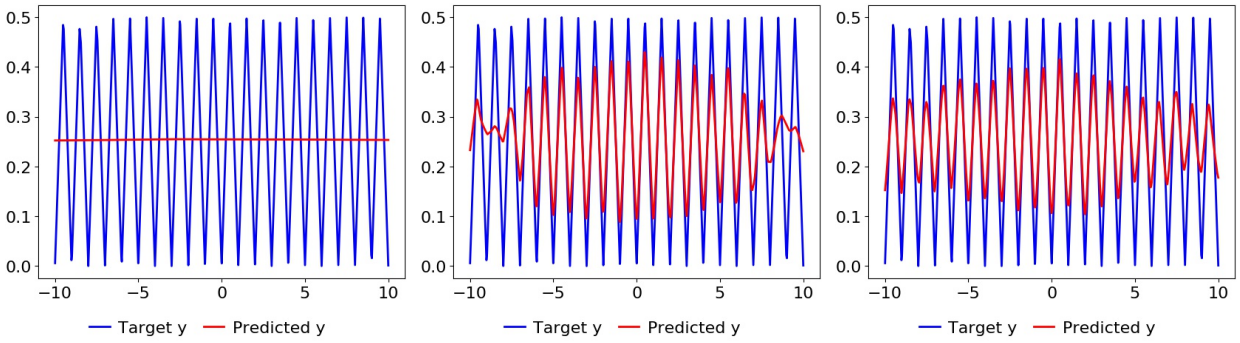
Figure 13: Scatter plots of 40 pairs of Wasserstein and neural distances, for $q = 2$. The underlying distributions are bivariate Gaussian distributions with 4 components. The red curve is the optimal parabolic fitting and LRE refers to the Least Relative Error. The red zone is the envelope obtained by stretching the optimal curve.

C Study of increasing group sizes for GroupSort networks



(a) Grouping size = 2 (b) Grouping size = 5 (c) Grouping size = 10

Figure 14: Reconstruction of a 20-piecewise linear function with varying grouping sizes ($k = 2, 5, 10$).



(a) Grouping size = 2 (b) Grouping size = 5 (c) Grouping size = 10

Figure 15: Reconstruction of a 40-piecewise linear function with varying grouping sizes ($k = 2, 5, 10$).

D Shared architecture for both GroupSort and ReLU networks

Operation	Feature Maps	Activation
$D(x)$		
Fully connected - q layers	width w	{GroupSort, ReLU}
Width w	{50}	
Depth q	{2, 4, 6, 8}	
Batch size	256	
Learning rate	0.0025	
Optimizer	Adam: $\beta_1 = 0.5$	$\beta_2 = 0.5$

Table 2: Hyperparameters used for the training of all neural networks