Approximating Lipschitz continuous functions with GroupSort neural networks

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

M. Sangnier
Sorbonne Université, CNRS, LPSM
Paris, France
maxime.sangnier@sorbonne-universite.fr

G. Biau
Sorbonne Université, CNRS, LPSM
Paris, France
gerard.biau@sorbonne-universite.fr

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).

One of the most natural ways to restrict the Lipschitz constants of neural networks is to limit their weights, as advocated by Arjovsky et al. (2017). This is however a delicate operation that may significantly affect the expressive power of the predictors. For example, Anil et al. (2019, Theorem 1) shows 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, introduced in Chernodub and Nowicki (2016).

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) and argue that the size of GroupSort networks can be significantly smaller.

(ii) Building on the work of Anil et al. (2019), we offer upper bounds on the depth and size of GroupSort neural networks that approximate 1-Lipschitz continuous functions on compact sets.

(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 piecewise linear functions with GroupSort neural networks. The approximation of Lipschitz continuous functions 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 \( \text{Lip}_1(E) \) the set of 1-Lipschitz real-valued functions on \( E \), i.e.,

\[
\text{Lip}_1(E) = \{ f : E \rightarrow \mathbb{R} : |f(x) - f(y)| \leq \|x - y\|, (x, y) \in E^2 \}.
\]

We let \( \mathcal{D} = \{ D_\alpha : \alpha \in \Lambda \} \) be the class of functions from \( \mathbb{R}^d \) to \( \mathbb{R} \) parameterized by feedforward neural networks of the form

\[
D_\alpha(x) = V_q \sigma(V_{q-1} \cdots \sigma(V_1 x + c_1) + c_2) + \cdots + c_{q-1} + c_q,
\]

where \( q \geq 2 \) and the characters below the matrices indicate their dimensions (lines \( \times \) columns). Thus, a network in \( \mathcal{D} \) has \((q - 1)\) hidden layers, and hidden layers from depth \( 1 \) to \((q - 1)\) are assumed to be of respective even widths \( v_1, \ldots, v_{q-1} \). 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 \( \Lambda \) of the functions in \( \mathcal{D} \).

With respect to the activations, we propose to use the GroupSort activation which separates the pre-activations into groups and sorts each group into ascending order. In particular, we use the GroupSort function with a grouping size equal to 2, applied on a given vector \( x_1, \ldots, x_{2n} \) as follows:

\[
\sigma(x_1, x_2, \ldots, x_{2n-1}, x_{2n}) = (\max(x_1, x_2), \min(x_1, x_2), \ldots, \max(x_{2n-1}, x_{2n}), \min(x_{2n-1}, x_{2n})).
\]

This activation is applied on pairs of components (which makes sense in (1) since the widths of the hidden layers are assumed to be even). GroupSort has been introduced in Chernodub and Nowicki (2016) as a 1-Lipschitz activation function that preserves the gradient norm of the input. 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, in the sense that \( \sigma(x, 0) = (\text{ReLU}(x), -\text{ReLU}(-x)) \), but the converse is not true. Consequently, GroupSort is expected to share some of the properties of ReLU networks such as, for example, parameterizing piecewise linear functions.

Throughout the manuscript, the notation \( \| \cdot \| \) (respectively, \( \| \cdot \|_\infty \)) means the Euclidean (respectively, the supremum) norm on \( \mathbb{R}^k \), with no reference to \( k \) as the context is clear. For \( W = (w_{i,j}) \) a matrix
of size $k_1 \times k_2$, we let $\|W\|_2 = \sup_{|x|_1=1} |Wx|$ be the 2-norm of $W$. Similarly, the $\infty$-norm of $W$ is $\|W\|_\infty = \sup_{|x|_1=1} |Wx|_\infty = \max_{i=1, \ldots, k_1} \sum_{j=1}^{k_2} |w_{ij}|$. We will also use the $(2, \infty)$-norm of $W$, i.e., $\|W\|_{2, \infty} = \sup_{|x|_1=1} |Wx|_\infty$. The following assumption plays a central role in our approach:

**Assumption 1.** For all $\alpha = (V_1, \ldots, V_q, c_1, \ldots, c_q) \in \Lambda$, $\|V_1\|_\infty \leq 1$, max$(\|V_2\|_\infty, \ldots, \|V_q\|_\infty) \leq 1$, and max$(\|c_i\|_\infty : i = 1, \ldots, q) \leq K_2$, where $K_2 \geq 0$ 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 $\mathcal{D} \subseteq \text{Lip}_1(\mathbb{R}^d)$.

Combining Lemma 1 with Arzelà-Ascoli theorem, it is easy to see that, under Assumption 1, the class $\mathcal{D}$ 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 $\mathcal{D}$ 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, which is arbitrarily close to $f$ with respect to the uniform norm. Our objective in the present article is to further explore the properties of these networks. We start in the next section by examining the case of piecewise linear functions.

## 3 Learning piecewise linear functions with GroupSort networks

The ability of feedforward ReLU neural networks to represent piecewise linear 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 $\lfloor \log_2(d+1) \rfloor$ (the symbol $\lfloor \cdot \rfloor$ 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 tackle the problem of representing piecewise linear functions with constrained GroupSort networks of the form (1).

### 3.1 Exact 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 ($m_f \geq 2$) if there exist a partition $\Omega = \{\Omega_1, \ldots, \Omega_{m_f}\}$ of $\mathbb{R}^d$ into polytopes and a collection $\ell_1, \ldots, \ell_{m_f}$ of affine functions such that, for all $x \in \Omega_i$, $i = 1, \ldots, m_f$, $f(x) = \ell_i(x)$.

At this stage no further assumption is made on the sets $\Omega_1, \ldots, \Omega_{m_f}$, which are just assumed to be polytopes in $\mathbb{R}^d$. An example of a piecewise linear function on the real line with $m_f = 4$ is depicted in Figure 1. As this figure suggests, the ambient space $\mathbb{R}^d$ can be further covered by a second partition $\tilde{\Omega} = \{\tilde{\Omega}_1, \ldots, \tilde{\Omega}_{M_f}\}$ of $M_f$ polytopes ($M_f \geq 1$), in such a way that the sign of the differences $\ell_i - \ell_{i'}$, $(i, i') \in \{1, \ldots, m_f\}^2$, does not change on the subsets $\tilde{\Omega}_1, \ldots, \tilde{\Omega}_{M_f}$. It is easy to see that the partition $\tilde{\Omega}$ is finer than $\Omega$ since, for each $i \in \{1, \ldots, M_f\}$ there exists $j \in \{1, \ldots, m_f\}$ such that $\tilde{\Omega}_j \subseteq \Omega_i$. This implies in particular that $M_f \geq m_f$. A visualization in dimension 1 of both $\Omega$ and $\tilde{\Omega}$ is given in Figure 1.

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

$$f = \max_{1 \leq k \leq M_f} \min_{i \in A_k} \ell_i,$$

(2)
where each $s_i$ is a non-empty subset of $\{1, \ldots, m_f\}$. 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, that justifies spending some time examining it.

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

Lemma 2 is an improvement of He et al. (2018, Lemma 5.1), which shows that $M_f \leq m_f!$. Our proof method exploits the inequality $M_f \leq C_{m_f(m_f-1)/2,d}$, 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 identity (2) is encapsulated in Lemma 3 below, which will be useful for later analysis, in combining maxima and minima in neural networks of the form (1).

**Lemma 3.** Let $f_1, \ldots, f_m: \mathbb{R}^d \rightarrow \mathbb{R}$ be a collection of functions ($m \geq 2$), each represented by a neural network of the form (1) with depth $q_i + 1$ and size $s_i$, $i = 1, \ldots, m$. Then there exist neural networks of the form (1) with depth at most $\max(q_1, \ldots, q_m) + \lceil \log_2(m) \rceil$ and size at most $s_1 + \cdots + s_m + 2m - 3$ that represent the functions $f = \max(f_1, \ldots, f_m)$ and $g = \min(f_1, \ldots, f_m)$.

In the specific case where $m = 2^n$ for some $n \geq 1$, then there exist neural networks of the form (1) with depth at most $\max(q_1, \ldots, q_m) + n$ and size at most $s_1 + \cdots + s_m + m - 1$ that represent $f$ and $g$.

It should be stressed that the use of GroupSort activations slightly reduces the size of the networks. Indeed, Arora et al. (2018, Lemma D.3), which is the analog of Lemma 3 with ReLU neural networks, asserts that the size is at most $s_1 + \cdots + s_m + 8m - 4$. By combining Lemma 2, Lemma 3, 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 $f \in \text{Lip}_1(\mathbb{R}^d)$ 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 at most $\lceil \log_2(M_f) \rceil + \lceil \log_2(m_f) \rceil + 1$ and its size is at most $3m_f M_f - M_f - 3$.

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 $O(d^{2m_f/M_f})$. 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). In a nutshell, 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 $\Omega$ satisfies some geometrical properties. Our next proposition examines the case where the sets $\Omega_1, \ldots, \Omega_{m_f}$ are convex.
**Proposition 1.** Let \( f \in \text{Lip}_1(\mathbb{R}^d) \) that is also \( m_f \)-piecewise linear with convex subdomains \( \Omega_1, \ldots, \Omega_m \). Then there exists a neural network of the form (1) verifying Assumption 1 that represents \( f \). Besides, its depth is at most \( 2\lceil \log_2(m_f) \rceil \) and its size is at most \( 3m_f^2 - m_f - 3 \).

Proposition 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, every connected subset is also convex.

**Proposition 2.** Let \( f \in \text{Lip}_1(\mathbb{R}) \) 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 at most \( 2\lceil \log_2(m_f) \rceil \) and its size is at most \( 3m_f^2 - m_f - 3 \).

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 at most \( \lceil \log_2(m_f) \rceil \) and its size is at most \( 3m_f^2 - 3 \).

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

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 smaller than \( m_f \). Note however that the obtained neural networks do not necessarily verify a requirement similar to the one of Assumption 1. One may consequently trade-off the boundedness of the weights for a smaller depth and size.

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

**Lemma 4.** Any neural network of the form (1) on the real line, with depth \( q \) and widths \( w_1, \ldots, w_{q-1} \), parameterizes a piecewise linear function with at most \( (w_1 + 1) \times \cdots \times w_{q-1} \) linear subdomains.

We deduce from this lemma that for a neural network of the form (1) with depth \( q \geq 2 \) and constant width \( w \), the maximum number of linear regions is \( O(w^{q-1}) \). Thus, as for ReLU neural networks (Montúfar et al., 2014), we see that the number of linear regions for GroupSort networks is likely to grow polynomially in \( w \) and exponentially in \( q \). Interestingly, Arora et al. (2018, Theorem 3.1) exhibits an upper bound on the number of linear regions for ReLU neural networks on the real line that is larger than for GroupSort networks. More precisely, this theorem shows that a ReLU network of depth \( q \) and constant width \( w \) will have at most \( 2^{q-1}w^{q-1} \) linear regions.

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

**Corollary 1.** Let \( f \in \text{Lip}_1(\mathbb{R}) \) that is also \( 2^{nk} \)-piecewise linear for some \( n \geq 1, k \geq 1 \). Then there exists a neural network of the form (1) verifying Assumption 1, with depth at most \( nk \) and size at most \( (2n - 1)k \), representing \( f \). On the opposite, any neural network of the form (1) verifying Assumption 1 and representing \( f \) with constant width and depth \( q \leq nk \), has a size at least \( (q - 1)2^{nk/(q-1)} \).

This theorem points out that when the depth \( q \) of the network is significantly reduced (e.g., \( q \ll nk \)), then the size increases exponentially. This result should be put in light with previous results regarding the expressiveness of deep neural networks (e.g., Telgarsky, 2015, 2016). In the same spirit, Eldan and Shamir (2016) show that there are simple functions expressible by a 3-layer feedforward neural network, which can be approximated by a 2-layer network only if the width is exponential in the dimension of the input.

### 4 Approximating Lipschitz continuous functions on compact sets

Following our plan, we tackle in this section the task of approximating Lipschitz continuous functions on compact sets using GroupSort neural networks. The space of continuous functions on \([0, 1]^d\) is
We start with the problem of learning a function $f$, 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 $\text{Lip}_1([0, 1]^d)$.

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

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. Close to our work is the one of Yarotsky (2017), who 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 $f \in \text{Lip}_1([0, 1]^d)$, there exists a ReLU neural network approximating $f$ with precision $\varepsilon$, 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 both depths are similar but ReLU networks are smaller in size. However, one has to keep in mind that 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 $d = 1$

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

In the context of ReLU networks, Yarotsky (2017, Theorem 2) shows that a depth of at most 6 is needed together with a size at most $\frac{60}{\varepsilon}$. Here again, we observe that GroupSort neural networks have a larger upper bound on the depth but a much smaller upper bound on the size. This is in line with our previous discussion on the trade-off between depth/size of neural networks (Lu et al., 2017; Raghu et al., 2017). Finally, for a 1-Lipschitz convex (or concave) function defined on $[0, 1]$, we have the following:

**Corollary 3.** Let $\varepsilon > 0$ and $f \in \text{Lip}_1([0, 1])$. Assume that $f$ is convex or concave. Then there exists a neural network $D$ of the form (1) verifying Assumption 1 such that $\|f - D\|_{\infty} \leq \varepsilon$. The depth of $D$ is at most $\lceil \log_2(1/\varepsilon) \rceil$ and its size is at most $2\lceil \log_2(1/\varepsilon) \rceil - 1$.

Interestingly, Yarotsky (2017, Proposition 2) concludes that the function $x^2$ on $[0, 1]$ can be approximated by a ReLU network with depth and size $O(\ln(1/\varepsilon))$, similarly to Corollary 3.

### 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.

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 $w = 60$. 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 2. 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 decreases 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.

Next, in a second series of experiments, we compare the performances of GroupSort networks against two baselines: ReLU neural networks without any constraint on the weights (which are known to be 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 \(\varepsilon\) 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.

The results are presented in Figure 3 (noises case) and Figure 4 (noisy case). We observe that in the noiseless setting, 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 networks 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 Figure 3c and Figure 4c that the number of linear regions for GroupSort neural networks is smaller than for ReLU networks. This corroborates the discussion following Lemma 4.

To finish this section, we finally motivate the use of GroupSort neural networks in the context of WGANs (Arjovsky et al., 2017). To do so, 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 $\mathcal{D}$ of the form (1) with a depth $q = 5$ and a constant width $w = 20$. As in the experiments above, we also implement ReLU neural networks with similar architecture (with and without constraints on the weights). For a pair of distributions $(\mu, \nu)$, our goal is to exemplify the relationship between the 1-Wasserstein distance $\sup_{f \in \text{Lip}^1(\mathbb{R}^2)} (E_{\mu} - E_{\nu})$ and the neural distance $\sup_{f \in \mathcal{D}} (E_{\mu} - E_{\nu})$ (Arora et al., 2017) computed over the class of functions $\mathcal{D}$.

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 using the Python package by Flamary and Courty (2017) and calculate the corresponding neural distance. Figure 5 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 GroupSort neural networks of the form (1). For the case where the depth is limited to $q = 2$, additional results are also provided in the Appendix. This shows the interest in using GroupSort networks as a principled training method for WGANs.

Figure 5: Scatter plots of 40 pairs of Wasserstein and neural distances, for $q = 5$. 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.

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. These properties open new perspectives for broader use of GroupSort networks.
particularly in the field of WGANs, where they were shown to provide promising approximation results for Wasserstein distances.

7 Broader impact

Recent years have crowned deep neural networks with, in some cases, billions of parameters causing the computational complexity to increase tremendously. We hope that the present article, which proposes new upper bounds on the depth and size of robust networks, will positively contribute to the literature interested in the mathematical properties of feedforward neural networks.

References

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

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

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

S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. Generalization and equilibrium in generative adversarial nets (GANs). In D. Precup and Y.W. Teh, editors, Proceedings of the 34th International Conference on Machine Learning, volume 70, pages 224–232. PMLR, 2017.

G. Biau, B. Cadre, M. Sangnier, and U. Tanielian. Some theoretical properties of GANs. The Annals of Statistics, in press, 2020.

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

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

A. Chernodub and D. Nowicki. Norm-preserving Orthogonal Permutation Linear Unit activation functions (OPLU). arXiv:1604.02313, 2016.

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

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

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

R. Eldan and O. Shamir. The power of depth for feedforward neural networks. In Conference on Learning Theory, pages 907–940, 2016.

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

R. Flamary and N. Courty. POT: Python Optimal Transport library, 2017. URL https://github.com/rflamary/POT

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

I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In International Conference on Learning Representations, 2015.
I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A.C. Courville. Improved training of Wasserstein GANs. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5767–5777. Curran Associates, Inc., 2017.

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

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

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

Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The expressive power of neural networks: A view from the width. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6231–6239. Curran Associates, Inc., 2017.

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

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

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

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

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

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

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

X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang. Improving the improved training of Wasserstein GANs: A consistency term and its dual effect. In *International Conference on Learning Representations*, 2018.

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

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

### A Technical results and complementary experiments

#### A.1 Proof of Lemma 1

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

10
Therefore, for \((x,y) \in (\mathbb{R}^d)^2\),
\[
\|f_1(x) - f_1(y)\|_\infty \leq \|V_1 x - V_1 y\|_\infty
\]
(since \(\sigma\) is 1-Lipschitz)
\[
= \|V_1(x-y)\|_\infty
\]
\[
\leq \|V_1\|_\infty \|x-y\|
\]
\[
\leq \|x-y\|
\]
(by Assumption 1).

Thus,
\[
\|f_2 \circ f_1(x) - f_2 \circ f_1(y)\|_\infty \leq \|V_2 f_1(x) - V_2 f_1(y)\|_\infty
\]
(since \(\sigma\) is 1-Lipschitz)
\[
\leq \|V_2\|_\infty \|f_1(x) - f_1(y)\|_\infty
\]
\[
\leq \|f_1(x) - f_1(y)\|_\infty
\]
(by Assumption 1)
\[
\leq \|x-y\|
\]

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

### A.2 Proof of Lemma 2

Recall that \(m_f \geq 2\). 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,
\[
\ell_i(x) - \ell_j(x) \leq 0 \iff x \cdot (a_i - a_j) \leq b_j - b_i.
\]

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 \((\ell_i, \ell_j)\), 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 \(C_{n,d} \leq (1 + n)^d\) becomes preferable (Devroye et al., 1996, Chapter 30). Thus, we have
\[
m_f \leq M_f \leq \min(2^{m_f^2/2}, (m_f/\sqrt{2})^{2d}).
\]

### A.3 Proof of Lemma 3

The proof follows the one of Arora et al. (2018, Lemma D.3) and uses induction on \(m\). The base case \(m = 2\) is clear using the GroupSort activation. For \(m > 2\), let \(g_1 = \max(f_1, \ldots, f_{[m/2]}\) and \(g_2 = \max(f_{[m/2]+1}, \ldots, f_m)\) (the symbol \([\cdot]\) stands for the integer part function). By the induction hypothesis, \(g_1\) and \(g_2\) can be represented by neural networks of the form (1) with depths at most \(\max(q_1, \ldots, q_{[m/2]} + \lceil \log_2([m/2]) \rceil)\) and \(\max(q_{[m/2]+1}, \ldots, q_m + \lceil \log_2([m/2]) \rceil)\), respectively, and sizes at most \(s_1 + \cdots + s_{[m/2]} + 2[m/2] - 3\) and \(s_{[m/2]+1} + \cdots + s_m + 2[m/2] - 3\), respectively.

Thus, using the same construction as in Anil et al. (2019, Theorem 3), the bivariate function \(G(x) = (g_1(x), g_2(x))\) can be implemented by a neural network of the form (1) with depth at most \(\max(q_1, \ldots, q_m + \lceil \log_2([m/2]) \rceil)\) and size \(s\) such that
\[
s \leq s_1 + \cdots + s_m + 2[m/2] + 2[m/2] - 6 + 1
\]
\[
= s_1 + \cdots + s_m + 2m - 5.
\]

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 at most \(\max(q_1, \ldots, q_m + \lceil \log_2(m) \rceil)\) and size at most \(s_1 + \cdots + s_m + 2m - 4 \leq s_1 + \cdots + s_m + 2m - 3\).
The second part of the theorem is shown by following a similar reasoning by induction. The case \( m = 2 \) is true since the function \( f = \max(f_1, f_2) \) can be represented by a neural network of the form (1) with depth \( q_1 + q_2 + 1 \) and size \( s_1 + s_2 + 1 \). Now, let \( m = 2^n \) with \( n > 1 \). We have that \( |m/2| = |m/2| = m/2 = 2^{n-1} \). By the induction hypothesis, \( g_1 \) and \( g_2 \) can be represented by neural networks of the form (1) with depths at most \( \max(q_1, \ldots, q_{m/2}) + n - 1 \) and size \( s_1 + \cdots + s_{m/2} + m/2 - 1 \), respectively, 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 at most \( \max(q_1, \ldots, q_m) + n - 1 \) 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 at most \( \max(k_1, \ldots, k_m) + n = \max(k_1, \ldots, k_m) + \log_2(m) \) and size at most \( s_1 + \cdots + s_m + m - 1 \).

### A.4 Proof of Theorem 1

Let \( f \in \text{Lip}_1(\mathbb{R}^d) \) 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). Combining (2) with Lemma 3, for each \( k \in \{1, \ldots, M_f\} \) there exists a neural network of the form (1), verifying Assumption 1 and representing the function \( \min_{i \in \mathbb{R}_k} \ell_i \), with depth at most \( \lceil \log_2(m_f) \rceil \) (since \( |s| \leq m_f \)) and size at most \( 3m_f - 3 \).

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

### A.5 Proof of Proposition 1

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

\[
f = \max_{1 \leq k \leq m_f} \min_{i \in \mathbb{N}_k} \ell_i.
\]

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 at most \( 2\lceil \log_2(m_f) \rceil \) and size at most \( 3m^2_f - m_f - 3 \).

### A.6 Proof of Proposition 2

Let \( f \in \text{Lip}_1(\mathbb{R}) \) that is also \( m_f \)-piecewise linear. The proof of the first statement is an immediate consequence of Proposition 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

\[
f = \max_{1 \leq k \leq m_f} \ell_k.
\]

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

The last claim of the proposition for \( m = 2^n \) is clear using Lemma 3.

### A.7 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 \( n = ([1/s])^d \) 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 \( \|f - \tilde{f}\|_\infty \leq \varepsilon \).
Using Proposition 1, we know that there exists a neural network of the form (1) verifying Assumption 1 and representing \( f \). Besides, its depth is at most \( 2 \lceil \log_2(3m_f) \rceil \) and its size is at most \( 3m_f^2 - m_f - 3 \). Consequently, we have that the depth of the neural network is \( O(d \log_2(\frac{2\sqrt{d}}{\epsilon})) \) and the size at most \( O(m^2) = O((\frac{2\sqrt{d}}{\epsilon})^{2d}) \).

A.8 Proof of Corollary 2

Let \( f \in \text{Lip}_1([0,1]) \) and \( f_m \) be the piecewise linear interpolation of \( f \) with the following \( 2^m + 1 \) breakpoints: \( k/2^m, k = 0, \ldots, 2^m \). We know that the function \( f_m \) approximates \( f \) with an error \( \epsilon_m \leq 2^{-m} \). In particular, for any \( m \geq \log_2(1/\epsilon) \), we have \( \epsilon_m \leq \epsilon \). 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 at most \( m \) and size at most \( 3 \times 2^m - 2^m - 3 \). Taking \( m = \lceil \log_2(1/\epsilon) \rceil \) shows the desired result.

A.9 Proof of Corollary 3

Let \( \epsilon > 0 \), let \( f \) be a convex (or concave) function in \( \text{Lip}_1([0,1]) \), and let \( f_m \) be the piecewise linear interpolation of \( f \) with the following \( 2^m + 1 \) breakpoints: \( k/2^m, k = 0, \ldots, 2^m \). The function \( f_m \) approximates \( f \) with an error \( \epsilon_m = 2^{-m} \). In particular, for any \( m \geq \log_2(1/\epsilon) \), we have \( \epsilon_m \leq \epsilon \). 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 at most \( m \) and size at most \( 2 \times 2^m - 1 \). Taking \( m = \lceil \log_2(1/\epsilon) \rceil \) leads to the desired result.

B Experiments: approximating Lipschitz functions

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

B.0.1 Piecewise linear regression function

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 6 and Figure 7) and in the model \( Y = f(X) + \epsilon \) (noisy case, see Figure 8 and Figure 9). Recall that in both cases, \( X \) follows a uniform distribution on \([-1.5,1.5]\) and the sample size is \( n = 100 \).

![Figure 6: 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).](image-url)
**B.0.2 Additional results for 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 10 is the case without noise while Figure 11 is the case with noise.
Figure 10: Reconstructing the function $f(x) = (1/15) \sin(15x)$ in the model $Y = f(X)$, with a dataset of size $n = 100$.

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

B.1 Experiments: calculating Wasserstein distances

Figure 12: 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.
### B.2 Network architecture and hyperparameters

| Operation       | Feature Maps | Activation |
|-----------------|--------------|------------|
| $D(x)$ Fully connected - $q$ layers | width $w$ | ReLU |
| Width $w$       | $\{20, 60\}$ |
| Depth $q$       | $\{2, 4, 6, 8\}$ |
| Batch size      | 32           |
| Learning rate   | 0.0025       |
| Optimizer       | Adam: $\beta_1 = 0.5$ $\beta_2 = 0.5$ |

Table 1: Hyperparameters used for the training of all neural networks