Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Communication Theory Group
Prof. Dr. H. Bölcskei

Semester Project in Pure Mathematics

Fall Semester 2018

Simon Gruening

Numerical Experiments on Deep Neural Network Function Approximation

Supervisor: Dmytro Perekrestenko

January 2019

Abstract

We explore how the theory of deep neural networks for function approximation intersects with the practical application of such networks. We develop a theory for sampling Gabor systems uniformly. We perform a numerical experiment to examine how well neural networks train with Leaky ReLU's in the hopes of reaching deeper networks. We examine the ability of neural networks to approximate basic functions in depth. Finally, we ascend to representation systems and examine the behaviour of networks approximating Gabor systems. We conclude that SGD/Adam alone is not strong enough to train deep networks. Adding skip connections to the architecture helps immensely. We were not able to demonstrate the efficacy of training with Leaky ReLU's. Finally, we examine how well a DNN is able to generalize to a Gabor system given its ability to approximate the generator function. We conclude that contrary to the theory, DNN's are not able to generalize very well, however that with increased depth and skip connections we are able to improve this ability drastically.

Contents

Notation

- All norms $\|\cdot\|$ are standard Euclidean unless specified otherwise.
- For some $f : \mathbb{R}^j \to \mathbb{R}^k$, the support $\operatorname{supp}(f) = \overline{\{x \in \mathbb{R}^j : f(x) = 0\}}$ includes the points in the closure.
- The natural numbers $\mathbb{N}$ include 0. We may restrict $\mathbb{R}$ to a specific range as $\mathbb{R}_{>a} := \{x \in \mathbb{R} : x > a\}$ for $a \in \mathbb{R}$.
- The centered open ball of dimension d and radius $r > 0$ is defined as $B_r^d(0) := \{x \in \mathbb{R}^d : \|x\| < r\} \subseteq \mathbb{R}^d$.
- With $|\cdot|$ we denote the absolute value, with $\#A$ the cardinality of a set A.
- The logarithm $\log(\cdot)$ denotes to the binary logarithm.
- We write for real or complex-valued functions f, g that $f \in \mathcal{O}(g)$ as $x \to \infty$ if $\exists M > 0, x_0 \in \mathbb{R}_{>0} : \forall x > x_0 : |f(x)| \leq Mg(x)$.
- A real-valued function f is in $\mathfrak{C}^N$ when it is N times continously differentiable.
- A sequence b indexed by i in $\mathbb{N} \cap [1, r]$ is denoted as $(b_i)_{i=1}^r$. Unless specified otherwise $b_i \in \mathbb{R}$ for all i and $\|b_i\|_\infty := \max_{i=1,\ldots,d} |b_i|$.
- The absolute maximum entry norm for a matrix A is denoted as $\|A\|_\infty := \max_{i,j} |A_{i,j}|$.
- Denote by $\Re f, \Im f$ the real and complex part respectively of a complex-valued function f.
- Let $\mathcal{U}(a, b)$ denote the uniform distribution on (a, b), $\mathcal{N}(0, 1)$ the standard normal distribution.

Chapter 1

Introduction

Neural networks are the current frontier of machine learning, with quickly advancing numerical results. The approximation results by Cybenko [3] and Hornik [4] for wide and shallow networks are very strong, however not very practical due to weak bounds on the overall size of the networks. We follow the premise in [1] of restricting width and varying depth instead. This allows for much better bounds in the dimensions of our networks.

First we adapt a practical definition of approximating "well". It is possible to draw parallels between information theory and the ability of neural networks to approximate, strengthening the foundation of this definition. We ascend from basic function approximation to representation systems. Given the ability to approximate a set of generating functions, we ask how well we can approximate functions this set can approximate. We hone in further to Gabor systems, and finally a Wilson Base by branching of into [2]. This is motivated by relevance in the Time-Frequency analysis of signals.

We shall develop a theory for sampling Gabor systems uniformly. We implement this sampling and wish to examine the practical application of our theory of approximating Gabor systems by experimentation in the field.

Chapter 2

Setting

We build the neural network solely from two components, the affine transformation and a nonlinearity.

We shall restrict the nonlinearities in our network to the recitfied linear unit:

Definition 2.1. *Define a function* $\mathrm{ReLU}\colon \mathbb{R} \to \mathbb{R}$ *as:*

$$\rho := \mathrm{ReLU}(x) := \begin{cases} x & \textit{if } x > 0, \\ 0 & \textit{else}. \end{cases}$$

Depending on context, we also define the element-wise d-dimensional ReLU*:*

$$\begin{aligned} \rho\colon \mathbb{R}^d &\to \mathbb{R}^d \\ (x_i)_{i=1}^d &\mapsto (\max(0, x_i))_{i=1}^d \end{aligned}$$

Definition 2.2. *Define an affine transformation $W\colon \mathbb{R}^{d_2} \to \mathbb{R}^{d_1}$ as:*

$$W(x) := Ax + b$$

with $A \in \mathbb{R}^{d_1 \times d_2}$, $b \in \mathbb{R}^{d_1}$

Definition 2.3. *For a fixed $W, L \in \mathbb{N}$, the width and number of layers of the neural network, and fixed $D_i, D_o \in \mathbb{N}$, the input and output dimension respectively, we define a neural network (or network) as*

$$\Phi(x) := W_L(\rho(W_{L-1}(\rho(...\rho(W_1(x))...))))$$

where W_i are affine transformations of the format

$$\begin{aligned} W_1 &:= A_1 x + b_1 && A \in \mathbb{R}^{W \times D_i}, b \in \mathbb{R}^W \\ W_{i=2,3,..,L-1} &:= A_i x + b_i && A \in \mathbb{R}^{W \times W}, b \in \mathbb{R}^W \\ W_L &:= A_L x + b_L && A \in \mathbb{R}^{D_o \times W}, b \in \mathbb{R}^{D_o} \end{aligned}$$

In the case $l = 1$ let:

$$W_1 := A_1 x + b_1 \qquad A \in \mathbb{R}^{D_o \times D_i}, b \in \mathbb{R}^{D_o}.$$

We call the entries in A_i, b_i the weights and biases of the network. Together we refer to the entries as parameters. In the numerical portion of the paper, the term weights shall include the biases.

Remark 2.4. *Note that the final affine transformation does not suffer a nonlinearity.*

Definition 2.5. *For a neural network Φ with width W, layers L, input and output dimensions D_i, D_o, we define*

- $\mathcal{W}(\Phi) := \max\{W, D_i, D_o\}$
- $\mathcal{L}(\Phi) := L$
- $\mathcal{B}(\Phi) := \max_{i=1,\ldots,L} \{\|A_i\|_\infty, \|b_i\|_\infty\}$
- $\mathcal{M}(\Phi) :=$ *number of non-zero entries in the parameters.*

Remark 2.6. *We shall work with fixed width neural networks for numerical ease, however most literature accomodates varied width layers. This will not make a difference since we can simply construct a network of maximal width for any varied width network, and set the superfluous parameters to 0. This will maintain connectivity, depth, and maximal weight size.*

Inspired by recent breakthroughs [5] [6], in order to train deeper networks we also define the skip connected network. Our blocksize shall be two layers, as this was one of the pioneered block sizes. Further, the identity can be constructed in ReLU networks with two layers, removing the need for a third layer.

Definition 2.7. *Define a* ReLU *block as:*

$$W_R(x) := \rho(W_2(\rho(W_1(x)))) + x$$

Compose these blocks as in Definition 2.3. We call this a skip connected neural network.

The first and last layer will not be part of a ReLU block. Notice that this implies all skip connected networks to have an even number of layers. The skip connection provides the training algorithm with an identity, furthermore the gradient can travel over the x and can reach deeper.

Remark 2.8. *Our theoretical networks contain no skip connections, however note that it is possible to construct the identity as a ReLU network as follows:*

$$\begin{tikzcd}
& \bullet \arrow[dr, "1"] & \\
\bullet \arrow[ur, "1"] \arrow[dr, "-1"'] & & \bullet \\
& \bullet \arrow[ur, "-1"'] &
\end{tikzcd}$$

It follows that for the simple skip connected network we may set the width of a new network as $2W + W = 3W$ and preserve our theory by simply running the identity parallel.

Hyperconnected networks are also proposed [5] [6], however in this case, the additional width would, with an analog construction, grow linearly with depth, this is unsavoury in view of our theory.

Definition 2.9. *For $L, M, D_i, D_o \in \mathbb{N}$, define:*

- $\mathcal{NN}_{L,M,D_i,D_o} :=$ { *all networks Φ with L or less layers, M or fewer connectivity, D_i input dimensions and D_o output dimensions* }
- $\mathcal{NN}_{\infty,M,D_i,D_o} := \bigcup_{L\in\mathbb{N}} \mathcal{NN}_{L,M,D_i,D_o}$
- $\mathcal{NN}_{L,\infty,D_i,D_o} := \bigcup_{M\in\mathbb{N}} \mathcal{NN}_{L,M,D_i,D_o}$
- $\mathcal{NN}_{\infty,\infty,D_i,D_o} := \bigcup_{L\in\mathbb{N}} \mathcal{NN}_{L,\infty,D_i,D_o}$

Definition 2.10. *For $L, M, d \in \mathbb{N}$, $R \in \mathbb{R}_{>0}$ define*

$$\mathcal{NN}^{R}_{L,M,d,1} := \{\Phi \in \mathcal{NN}_{L,M,d,1} : \mathcal{B}(\Phi) \leq R\}$$

Definition 2.11. *For a $m \in \mathbb{N}$, $\varepsilon \in (0,\infty)$, we say a network Φ has (m,ε)-quantized weights if all weights are in $2^{-m\lceil \log(\frac{1}{\varepsilon})\rceil}\mathbb{Z} \cap [-\varepsilon^{-m}, \varepsilon^{-m}]$.*

Chapter 3

Theory

3.1 Approximation of Basic Functions

We adapt the relevant theory from [1] to our context. We begin by approximating x^2 and xy with neural networks, and finally any polynomial in lieu of the infinity norm. With Weierstrass we then make the jump to any continous function. Due to it's prevalence in signal theory, we examine $cos(x)$ next.

The following proposition builds on Yarotsky's sawtooth construction [7].

Proposition 3.1. *[1] There exists a $C > 0$ such that for any $\frac{1}{2} > \varepsilon > 0$, there exists a neural network Φ_ε with $D_i = D_o = 1$ satisfying:*

- $\mathcal{L}(\Phi_\varepsilon) \leq C \log(\frac{1}{\varepsilon})$
- $\mathcal{W}(\Phi_\varepsilon) = 4$
- $\mathcal{B}(\Phi_\varepsilon) \leq 4$
- $\Phi_\varepsilon(0) = 0$
- $\|\Phi_\varepsilon(x) - x^2\|_{L^\infty([0,1])} \leq \varepsilon$

Through the identity $xy = \frac{1}{2}((x+y)^2 - x^2 - y^2)$ and linear combination of neural networks we receive:

Proposition 3.2. *[1] There exists a $C > 0$ such that for any $D \in \mathbb{R}_{>0}$, $\frac{1}{2} > \varepsilon > 0$ there exists a neural network Φ_ε with $D_i = 2$, $D_o = 1$ satisfying:*

- $\mathcal{L}(\Phi_{D,\varepsilon}) \leq C \log(\lceil D \rceil^2 \frac{1}{\varepsilon})$

- $\mathcal{W}(\Phi_{D,\varepsilon}) \leq 12$
- $\mathcal{B}(\Phi_{D,\varepsilon}) \leq \max(4, 2\lceil D \rceil^2)$
- $\forall x \in \mathbb{R} : \Phi_{D,\varepsilon}(x, 0) = \Phi_{D,\varepsilon}(0, x) = 0$
- $\forall x, y \in \mathbb{R} : \|\Phi_{D,\varepsilon}(x, y) - xy\|_{L^\infty([0,1])} \leq \varepsilon$

Through iterative construction we are now able to approximate x^m. We may retrieve the individual monomials of a polynomial in such a construction by taking the output at a layer respective to the degree d of a monomial x^d, and running it parallel to the calculation of x^m, multiplying it in accordance to some coefficient.

Theorem 3.3. *[1] There exists a $C > 0$ such that for any $D, A \in \mathbb{R}_{>0}$, $\frac{1}{2} > \varepsilon > 0$, and polynomial $p_m(x) = \sum_{i=0}^{m} a_i x^i$ of degree m with weights bounded absolutely by A: $\forall i : |a_i| \leq A$, there exists a neural network $\Phi_{p_m,D,\varepsilon}$ with $D_i = D_o = 1$ satsfying:*

- $\mathcal{L}(\Phi_{p_m,D,\varepsilon}) \leq Cm[\log(\lceil A \rceil) + \log(\frac{1}{\varepsilon} + m\log(\lceil D \rceil) + \log(m)]$
- $\mathcal{W}(\Phi_{p_m,D,\varepsilon}) \leq 16$
- $\mathcal{B}(\Phi_{p_m,D,\varepsilon}) \leq \max(A, 8\lceil D \rceil^{2m-2})$
- $\|\Phi_{p_m,D,\varepsilon} - p_m\|_{L^\infty([-D,D])} \leq \varepsilon$

Through use of the MacLaurin series representation, previous theorems, and symmetries inherent in $\cos$ we are able to prove the following theorem:

Theorem 3.4. *[1] There exists a $C > 0$ such that for any $D, a \in \mathbb{R}_{>0}$, $\frac{1}{2} > \varepsilon > 0$, there exists a neural network $\Psi_{a,D,\varepsilon}$ with $D_i = D_o = 1$ satisfying:*

- $\mathcal{L}(\Psi_{a,D,\varepsilon}) \leq C[(\log(\frac{1}{\varepsilon})^2 + \log(\lceil aD \rceil)]$
- $\mathcal{W}(\Psi_{a,D,\varepsilon}) \leq 16$
- $\mathcal{B}(\Psi_{a,D,\varepsilon}) \leq C$
- $\|\Psi_{a,D,\varepsilon}(x) - \cos(ax)\|_{L^\infty([-D,D])} \leq \varepsilon$

3.2 Approximation by Representation Systems

Having laid the ground work, we may now expand to much stronger theory. It is possible to show that being able to approximate a generator function, neural networks are further able to approximate functions generated by this function in the sense presented below.

3.2.1 Notions of Effective Approximation

We follow an abbreviated schedule of definitions from [1].

Definition 3.5. *Fix some $d \in \mathbb{N}$. Define a function class $\mathcal{C}$ as a compact set of functions in $L^2(\Omega)$ for some $\Omega \subseteq \mathbb{R}^d$.*

Definition 3.6. *A representation system is a $\mathcal{D} := (\varphi_i)_{i \in I} \subseteq L^2(\Omega)$ for some countable index I.*

Throghout this section, fix a $d \in \mathbb{N}$, $\Omega \subseteq \mathbb{R}^d$, a function class $\mathcal{C}$, and a representation system $\mathcal{D}$.

Definition 3.7. *For any $l \in \mathbb{N}$, we call*

$$\mathfrak{E}^l := \{E : \mathcal{C} \to \{0,1\}^l\}$$

the binary encoders, and

$$\mathfrak{D}^l := \{D : \mathcal{C} \to \{0,1\}^l\}$$

the binary decoders of length l.

Definition 3.8. *A binary decoder and encoder achieve uniform error ε over $\mathcal{C}$ if:*

$$\sup_{f \in \mathcal{C}} \|D(E(f)) - f\|_{L^2(\Omega)} \leq \varepsilon$$

Definition 3.9. *The minimax code length for some $\varepsilon > 0$ is defined as*

$$L(\varepsilon, \mathcal{C}) := \min\{l \in \mathbb{N} : \exists (E, D) \in \mathfrak{E}^l \times \mathfrak{D}^l : (E, D) \text{ achieve uniform error } \varepsilon \text{ over } \mathcal{C}\}$$

Definition 3.10. *We call*

$$\gamma^*(\mathcal{C}) := \sup\{\gamma \in \mathbb{R} : L(\varepsilon, \mathcal{C}) \in \mathcal{O}(\varepsilon^{-1/\gamma}), \varepsilon \to 0\}$$

the optimal exponent.

The optimal exponend describes the "resolution" of a function class, or the rate of additional bits required when asking for a more detailed description of the function class. This definition is fundamental, and as such its link to Kolmogorov entropy is not surprising [8] [9].

We now would like to examine systems of linear combinations of our representation system under L^2 error. This in itself however is not very well suited to numerical experimentation, and so we seek to limit the search space to a more manageable degree. We achieve this in limiting the number of participating functions polynomially, and their coefficients uniformly.

Definition 3.11. *The effective best M-term approximation error of f in $\mathcal{D}$ respective of some global polynomial π and $D > 0$ is defined as:*

$$\widetilde{\Gamma}^{\mathcal{D}}_{M,\pi,D}(f) := \inf_{\substack{I_M \subseteq \{1,2,\dots,\pi(M)\} \\ \#I_M = M, (c_i)_{i \in I_M} \text{ in } \mathbb{R}, \max_{i \in I_M} |c_i| \leq D}} \left\| f - \sum_{i \in I_M} c_i \varphi_i \right\|_{L^2(\Omega)}$$

Definition 3.12. *The effective best M-term approximation rate of $\mathcal{C}$ in the representation system D is defined as:*

$$\gamma^{*,eff}(\mathcal{C}, \mathcal{D}) := \\ \sup\{\gamma > 0 : \exists D > 0, \pi \text{ polynomial } : \sup_{f \in \mathcal{C}} \widetilde{\Gamma}^{\mathcal{D}}_{M,\pi,D}(f) \in \mathcal{O}(M^{-\gamma}), M \to \infty\}$$

Theorem 3.13. *[10] [9] $\gamma^{*,eff}(\mathcal{C}, \mathcal{D}) \leq \gamma^*(\mathcal{C})$.*

This powerful theorem lends meaning to the subsequent definition.

Definition 3.14. *A function class $\mathcal{C}$ is named optimally representable by $\mathcal{D}$ if*

$$\gamma^{*,eff}(\mathcal{C}, \mathcal{D}) = \gamma^*(\mathcal{C})$$

Analogously to linear systems, we may introduce this measure of approximation for neural networks.

Definition 3.15. *We call*

$$\gamma^{*,eff}_{\mathcal{NN}}(\mathcal{C}) := \\ \sup\{\gamma > 0 : \exists \pi \text{ polynomial } : \sup_{f \in \mathcal{C}} \inf_{\Phi \in \mathcal{NN}^{\pi(M)}_{\pi(\log(M)),M,d,1}} \|f - \Phi\|_{L^2(\Omega)} \in \mathcal{O}(M^{-\gamma}), M \to \infty\}$$

the effective best M-weight approximation rate of $\mathcal{C}$ by neural networks.

A lengthy and involved proof provides us with a result analog to 3.13. The existence of this parallel is striking, as the medium being inspected is very different.

Theorem 3.16. *[1] $\gamma^{*,eff}_{\mathcal{NN}}(\mathcal{C}) \leq \gamma^*(\mathcal{C})$*

This once again leads meaning to a definition of optimality:

Definition 3.17. *We say that a function class C is optimally representable by neural networks if*

$$\gamma^{*,eff}_{\mathcal{NN}}(\mathcal{C}) = \gamma^*(\mathcal{C})$$

Definition 3.18. *We call $\mathcal{D}$ effectively representable by neural networks if there exists a bivariate polynomial π such that $\forall i \in \mathbb{N}, 0 < \varepsilon < \frac{1}{2} : \exists \Phi_{i,\varepsilon} \in \mathcal{NN}_{\infty,\infty,d,1}$:*

- $\mathcal{M}(\Phi_{i,\varepsilon}) \leq \pi(\log(\frac{1}{\varepsilon}), \log(i))$
- $\mathcal{B}(\Phi_{i,\varepsilon}) \leq \pi(\frac{1}{\varepsilon}, i)$
- $\|\varphi_i - \Phi_{i,\varepsilon}\|_{L^2(\Omega)} \leq \varepsilon$

3.2.2 Representation Systems

We link now the connectivity of neural networks with the idea of best M-term approximation rate in representation systems. The following result is central.

Theorem 3.19. *[1] If the representation system $\mathcal{D} = (\varphi_i)_{i\in\mathbb{N}}$ is effectively representable by neural networks, then for any $0 < \gamma < \gamma^{*,eff}(\mathcal{C}, \mathcal{D})$ there exists polynomial π , function $\Psi : (0, \frac{1}{2}) \times \mathcal{C} \to \mathcal{NN}_{\infty,\infty,d,1}$ such that for any $f \in \mathcal{C}, 0 < \varepsilon < \frac{1}{2}$:*

- *$\Psi(\varepsilon, f)$ has $(\lceil \pi(\log(\frac{1}{\varepsilon})) \rceil, \varepsilon)$-quantized weights*
- $\|f - \Psi(\varepsilon, f)\|_{L^2\Omega} \leq \varepsilon$
- $\mathcal{L}(\Psi(\varepsilon, f)) \leq \pi(\log(\frac{1}{\varepsilon}))$
- $\mathcal{B}(\Psi(\varepsilon, f)) \leq \pi(\frac{1}{\varepsilon})$
- $\mathcal{M}(\Psi(\varepsilon, f)) \in \mathcal{O}(\varepsilon^{-1/\gamma}), \varepsilon \to 0$

In preparation of the next section, we cite the following proposition on linear combinations of translations of a given generator function. Its proof follows from explicitly performing canonincal linear combination on networks.

Proposition 3.20. *[1] Fix $r, d \in \mathbb{N}$, $p \in \mathbb{R}_{\geq 1} \cup \{\infty\}$, and $f \in L^p(\mathbb{R}^d)$.*

If there exists a bivariate polynmial π such that for all $D \geq 1, 0 < \varepsilon < \frac{1}{2}$ there is a network $\Phi_{D,\varepsilon} \in \mathcal{NN}_{\infty,\infty,d,1}$ with:

- $\mathcal{M}(\Phi_{D,\varepsilon}) \leq \pi(\log(\frac{1}{\varepsilon}), \log(D))$
- $\|f - \Phi_{D,\varepsilon}\|_{L^p([-D,D]^d)} \leq \varepsilon$

Then there exists a polynomial $\tilde{\pi}$ such that for all $c = (c_i)_{i=1}^r \subseteq \mathbb{R}$, $b = (b_i)_{i=1}^r \subseteq \mathbb{R}$, $E \geq 1$, and $0 < \eta < \frac{1}{2}$ there exists a network $\Phi_{c,b,E,\eta} \in \mathcal{NN}_{\infty,\infty,d,1}$ with

- *$\mathcal{M}(\Phi_{c,b,E,\eta}) \leq \tilde{\pi}(\log(\frac{1}{\eta_c}), \log(E_b))$ where*
  $E_b = \lceil E + \max_{i=1,\dots,r} \|b_i\|_\infty \rceil$ *and* $\eta_c = \frac{\eta}{\max\{1, \sum_{i=1}^r |c_i|\}}$

- $\left\| \sum_{i=1}^{r} c_i f(x - b_i) - \Phi_{c,b,E,\eta})(x) \right\|_{L^p([-E,E]^d)} \leq \eta$

Further, if the weights of the networks $\Phi_{D,\varepsilon}$ are polynomially bounded in D, $\frac{1}{\varepsilon}$, then

- *The weights of the networks $\Phi_{c,b,E,\eta}$ are polynomially bounded in*

$$\max\left\{1, \sum_{i=1}^{r} |c_i|\right\}, \max\left\{1, \max_{i=1,\dots,r} \|b_i\|_\infty\right\}, E, \frac{1}{\eta}$$

Note in particular that we may preserve the polynomial boundedness of weights.

3.2.3 Gabor Systems to Neural Networks

We now inspect a specific representation system, the Gabor System. The field of Time-Frequency Analysis motivates the study of the time and frequency domain of signals simultaneously. The Gabor System lends itself well to this purpose, being generated by the modulations and translations of a generator function. We are able to loop back to neural networks at the end of the section.

Definition 3.21. *Let $d \in \mathbb{N}_{>0}, f \in L^2(\mathbb{R}^d), x, \xi \in \mathbb{R}^d$. Define*

$$\begin{aligned} T_x : L^2(\mathbb{R}^d) &\to L^2(\mathbb{R}^d) \\ f(t) &\mapsto f(t - x) \end{aligned}$$

and

$$\begin{aligned} M_\xi : L^2(\mathbb{R}^d) &\to L^2(\mathbb{R}^d, \mathbb{C}) \\ f(t) &\mapsto e^{2\pi i \langle \xi, t \rangle} f(t) \end{aligned}$$

Let further $\Omega \subseteq \mathbb{R}^d$, $\alpha, \beta \in \mathbb{R}_{>0}$, and the generating function be $g \in L^2(\mathbb{R}^d)$. The Gabor System is then defined as the set:

$$\mathcal{G}(g, \alpha, \beta, \Omega) := \{M_\xi T_x g|_\Omega \in L^2(\Omega) : (x, \xi) \in \alpha\mathbb{Z}^d \times \beta\mathbb{Z}^d\}$$

The translation operator is not very interesting in the context of affine transformations, however the modulation operator provides a larger obstacle. The following Lemma follows by a constructive proof based on Euler's formula, Theorem 3.4, and Proposition 3.2.

Lemma 3.22. *[1] Fix $d \in \mathbb{N}$, generator function $f \in L^2(\mathbb{R}^d) \cap L^\infty(\mathbb{R}^d)$. Assume that for any $D \in \mathbb{R}_{\geq 1}, 0 < \varepsilon < \frac{1}{2}$ there exists a network $\Phi_{D,\varepsilon} \in \mathcal{NN}_{\infty,\infty,d,1}$ such that:*

$$\|f - \Phi_{D,\varepsilon}\|_{L^\infty([-D,D]^d)} \leq \varepsilon$$

Then there exists a $C \in \mathbb{R}_{>0}$ such that for any $D \in \mathbb{R}_{\geq 1}, 0 < \varepsilon < \frac{1}{2}, \xi \in \mathbb{R}^d$ there exist networks $\Phi^{Re}_{D,\xi,\varepsilon}, \Phi^{Im}_{D,\xi,\varepsilon} \in \mathcal{NN}_{\infty,\infty,d,1}$ such that:

- $\mathcal{L}(\Phi^{Re}_{D,\xi,\varepsilon}), \mathcal{L}(\Phi^{Im}_{D,\xi,\varepsilon}) \leq C[(\log(\frac{1}{\varepsilon}))^2 + \log(D) + \log(\lceil \|\xi\|_\infty \rceil)] + \mathcal{L}(\Phi_{D,\varepsilon})$
- $\mathcal{M}(\Phi^{Re}_{D,\xi,\varepsilon}), \mathcal{M}(\Phi^{Im}_{D,\xi,\varepsilon}) \leq C[(\log(\frac{1}{\varepsilon}))^2 + \log(D) + \log(\lceil \|\xi\|_\infty \rceil)] + \mathcal{M}(\Phi_{D,\varepsilon})$
- $\|\Re(\mathcal{M}_\xi f) - \Phi^{Re}_{D,\xi,\varepsilon}\|_{L^\infty([-D,D]^d)} + \|\Im(\mathcal{M}_\xi f) - \Phi^{Im}_{D,\xi,\varepsilon}\|_{L^\infty([-D,D]^d)} \leq 3\varepsilon$

Further, if the weights of ther $\Phi_{D,\varepsilon}$ are polynomially bounded in D and $\frac{1}{\varepsilon}$, then the weights of $\Phi^{Re}_{D,\xi,\varepsilon}, \Phi^{Im}_{D,\xi,\varepsilon}$ are polynomially bounded in D, $\frac{1}{\varepsilon}$, and $\|\xi\|_\infty$.

And so, if our generator function can be approximated well by a neural network, we may also approximate any modulation. We are able to expand this to the effective representation of the entire Gabor system.

Remark 3.23. *Note that it is possible to order the elements of $\mathcal{G}(g, \alpha, \beta, \Omega)$ as $\varphi_i = M_{\xi_i} T_{x_i} g|_\Omega$ such that $\|x_i\|_\infty, \|\xi_i\|_\infty$ do not grow faster than polynomial in i [1].*

We restrict to the assumption that Ω is bounded. In order to transition from $\|\cdot\|_{L^\infty(\Omega)}$ to $\|\cdot\|_{L^2(\Omega)}$, the following proof makes use of the fact that:

$$\|f\|_{L^2(\Omega)} = \left(\int\limits_\Omega |f(x)|^2 dx \right)^{\frac{1}{2}} \leq \left(\int\limits_\Omega \|f\|^2_{L^\infty(\Omega)} dx \right)^{\frac{1}{2}} = |\Omega|^{\frac{1}{2}} \|f\|_{L^\infty(\Omega)}$$

Combining this with Proposition 3.20, Remark 3.23, and Lemma 3.22 we receive:

Theorem 3.24. *[1] Fix $d \in \mathbb{N}, \alpha, \beta \in \mathbb{R}_{>0}$, and $\Omega \subseteq \mathbb{R}^d$ bounded. Fix a generator function $g\ inL^2(\mathbb{R}^d) \cap L^\infty(\mathbb{R}^d)$. If there exists a polynomial π such that for any $0 < \varepsilon < \frac{1}{2}$ there exists a network $\Phi_\varepsilon \in \mathcal{NN}_{\infty,\infty,d,1}$ such that:*

- $\mathcal{M}(\Phi_\varepsilon) \leq \pi(\log(\frac{1}{\varepsilon}))$
- $\mathcal{B}(\Phi_\varepsilon) \leq \pi(\frac{1}{\varepsilon})$
- $\|g - \Phi_\varepsilon\|_{L^\infty(\Omega)} \leq \varepsilon$

then the Gabor system $\mathcal{G}(g, \alpha, \beta, \Omega)$ is effectively representable by neural networks.

In conclusion, with the previous result and Theorem 3.19, we may now establish the link of approximating function classes between the Gabor system and neural networks:

Theorem 3.25. *[1] Fix $d \in \mathbb{N}, \alpha, \beta \in \mathbb{R}_{>0}$, and $\Omega \subseteq \mathbb{R}^d$ bounded. Fix a generator function $g\, in L^2(\mathbb{R}^d) \cap L^\infty(\mathbb{R}^d)$. If there exists a polynomial π such that for any $0 < \varepsilon < \frac{1}{2}$ there exists a network $\Phi_\varepsilon \in \mathcal{NN}_{\infty,\infty,d,1}$ such that:*

- $\mathcal{M}(\Phi_\varepsilon) \leq \pi(\log(\frac{1}{\varepsilon}))$
- $\mathcal{B}(\Phi_\varepsilon) \leq \pi(\frac{1}{\varepsilon})$
- $\|g - \Phi_\varepsilon\|_{L^\infty(\Omega)} \leq \varepsilon$

then for any function class $\mathcal{C}$ we have that:

$$\gamma_{\mathcal{NN}}^{*,eff} \geq \gamma^{*,eff}(\mathcal{C}, \mathcal{G}(g, \alpha, \beta, \Omega)).$$

Remark 3.26. *It follows that if a function class $\mathcal{C}$ is optimally representable by the representation system $\mathcal{D} = \mathcal{G}(g, \alpha, \beta, \Omega)$, then it is optimally representable by neural networks.*

3.2.4 Wilson Basis

Further specifying, we inspect a Wilson Basis. This is included under the Gabor system [2]. We lose some generality, none of the approximative strength, and gain availability for numerical experimentation.

Definition 3.27. *Given a real-valued compactly support generator function $g \in \mathfrak{C}^N$ for some $N \in \mathbb{N} \cup \{\infty\}$, we define our specific Wilson Basis as the set of all:*

$$\Psi_{k,l}(x) := \begin{cases} g(x-k) & \text{if } l = 0, \\ \sqrt{2}\cos(2\pi l x) g(x - \frac{k}{2}) & \text{if } l + k \text{ even}, \\ \sqrt{2}\sin(2\pi l x) g(x - \frac{k}{2}) & \text{if } l + k \text{ odd} \end{cases}$$

for $k \in \mathbb{Z}, l \in \mathbb{N}$.

Theorem 3.28. *[11] The system in definition 3.27 forms an orthonormal basis for $L^2(\mathbb{R})$.*

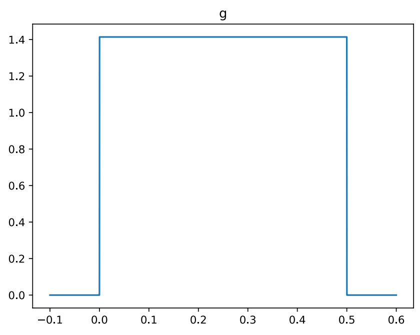
Remark 3.29. *We examine a polynomial-derived generator function g from [2] inspired by the constructive process in [12]:*

$$\theta(x) := \begin{cases} \frac{\pi}{4} & \textit{if } x \geq \frac{1}{4}, \\ -\frac{\pi}{4} & \textit{if } x \leq -\frac{1}{4}, \\ 96\pi x^5 - 20\pi x^3 + \frac{15}{8}\pi x & \textit{else} \end{cases}$$

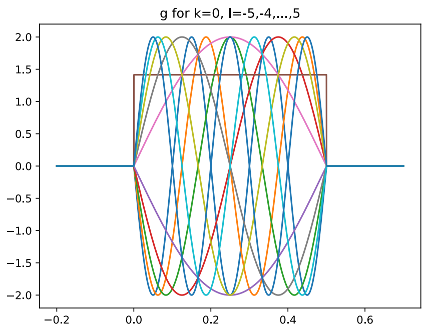
$$s(x) := \sin\left(\theta(x) + \frac{\pi}{4}\right)$$

$$c(x) := \cos\left(\theta(x) + \frac{\pi}{4}\right)$$

$$g(x) := \sqrt{2}s\left(x + \frac{1}{4}\right)c\left(x - \frac{1}{4}\right)$$



(a) Unmodulated, non-translated g



(b) See $\Psi_{k,l}(x)$ in 3.27

Remark 3.30. *Note that numerically, this function looks almost like the indicator function $\chi_{[0,0.5]}$.*

3.3 Random Sampling of a Gabor System

Recall the definition of a Gabor system 3.21. We wish to find a way to sample x, ξ uniformly in order to examine the system fairly.

3.3.1 Preliminaries

For the purpose of simulation we restrict $\Omega := \overline{B_D^d(0)} \subseteq \mathbb{R}^d$ for some $D \in \mathbb{R}_{>0}$, $d \in \mathbb{N}$. Further fix a generator $g : \mathbb{R}^d \to \mathbb{R} \in L^2(\mathbb{R}^d)$ such that $g|_\Omega \in L^2(\Omega) \cap \mathfrak{C}_c^0(\Omega)$ is compactly supported and $g|_{\mathbb{R}^d \setminus \Omega} \equiv 0$. Further let $\alpha, \beta \in \mathbb{Q}_{>0}$. By the limitations of the real world this must be, and by density we do not lose precision. We will benefit from the periodicity of modulation and boundedness of Ω to restrict our sample space.

We have that:

$$\begin{aligned} M_\xi g(t) &= e^{2\pi i \langle \xi, t \rangle} g(t) \\ &= \cos(2\pi\langle \xi, t\rangle) g(t) + i \sin(2\pi \langle \xi, t \rangle) g(t) \end{aligned}$$

is a periodic function. Thus it is sufficient to sample ξ such that $\langle \xi, t \rangle \in [0, 1)$, and by extension of symmetry we may also sample ξ uniformly from $[-1, 1]$. Note that Lemma 3.22 creates a seperate network for the $\Re$ and $\Im$ components, and so we may restrict ourselves to only a single component while maintaining the theoretical bounds.

Lemma 3.31. $\forall \xi \in \mathbb{R}^d : \left[\|\xi\| \leq \dfrac{1}{D} \iff \forall t \in \overline{B_D^d(0)} : |\langle \xi, t \rangle| \leq 1 \right]$.

Proof. [1] With an application of the Cauchy-Schwarz inequality we find:

$$t \in \overline{B_D^d(0)} \text{ a scalar multiple of } \xi \implies |\langle \xi, t\rangle| = \|t\| \cdot \|\xi\|.$$

Choose $t \in B_D^d(0)$ maximally as $\|t\| = D$, linearly dependent on ξ , to find:

$$\|\xi\| = \frac{|\langle \xi, t \rangle|}{\|t\|} \leq \frac{1}{D}.$$

Conversely we find that:

$$\|\xi\| \leq \frac{1}{D} \implies \langle \xi, t \rangle \leq \|\xi\| \|t\| \leq \frac{1}{D} D \leq 1.$$

□

We may thus restrict our sample space to $\xi \in \overline{B_{D^{-1}}^d(0)}$ uniformly, then snap it to the nearest point on the Lattice.

Notice for the sampling of translation first that by Heine-Borel:

$$\text{supp}(g) \text{ compact} \implies \text{supp}(g) \text{ bounded}$$

[1]Thank you to Severin Schraven for assisting me in the sketch of this proof

Further since $\forall z \in \mathbb{C} : e^z \neq 0$, and since our support is compact and thus is equal to the closure of the vanishing points we find:

$$\text{supp}(g) = \text{supp}(M_\xi g)$$

From the assumption of $\text{supp}(g) \subseteq \overline{B^d_D(0)}$ we have:

$$\forall x \in \mathbb{R}^d : \|x\| > 2D \implies T_x g(t) = T(t - x) \equiv 0.$$

As we can represent the constant 0 function with a single ReLU, we may neglect the cases of $T_x g \equiv 0$, for which the support has been shifted outside of Ω. To ensure that our sample is indeed uniform and does not oversample the constant 0 function, we enforce the more strict condition $\text{supp}(g) = \overline{B^d_D(0)}$.

We can now finally sample the Gabor System:

$$(x, \xi) \in (\overline{B^d_{2D}(0)}, \overline{B^d_{D^{-1}}(0)}).$$

3.3.2 Sampling Uniformly from the Ball

We make use of the method noted by Muller [13] for uniform n-Sphere sampling. For the open n-Ball $B^n_r(0)$ we can sample as follows:

- $V_1, ..., V_n \overset{i.i.d}{\sim} \mathcal{N}(0, 1)$
- $S := \sum_{i=1}^n V_i^2$
- $x := \left(\frac{V_1}{\sqrt{S}}, ..., \frac{V_n}{\sqrt{S}}\right) \in S^1_n(0)$
- $x_f := xr \in B^n_r(0)$ for $r \in \mathcal{U}(0, r)$

This does not require Monte-Carlo, which slows dramatically in higher dimensions.

3.3.3 1-Dimensional Case

The problem now presents itself in making sure our uniform sample is on the lattice. We first specialize now to dimension $d = 1$, $D \in \mathbb{Q}_{>0}$, so that $g \in L^2$ with $\text{supp}(g) = [-D, D]$. Our generator g must also satisfy our previous assumptions in 3.3. We see that in this case we may accomplish sampling from the lattice by simple rounding. We sample:

$$(x, \xi) \in \mathcal{U}([-2D, 2D]) \times \mathcal{U}\left(\left[-\frac{1}{D}, \frac{1}{D}\right]\right)$$

In the 1-dimensional case we can pick $\alpha, \beta \in \mathbb{Q}_{>0}$ such that:

$$\frac{2D}{\alpha} \in \mathbb{N}_{>0} \implies 2D \in \alpha\mathbb{Z}$$
$$\frac{1}{D\beta} \in \mathbb{N}_{>0} \implies \frac{1}{D} \in \beta\mathbb{Z}$$

This choice avoids a lattice dense in Ω after restricting our sample space modulo $\frac{1}{D}$ for ξ. When rounding our sample (x, ξ) to the nearest point in the lattice $\alpha\mathbb{Z} \times \beta\mathbb{Z}$, we are neglecting the two points denoting the boundary of the interval by the probability of an interval $\frac{\alpha}{2}$ and $\frac{\beta}{2}$ respectively. For a fair choice we thus instead sample from:

$$(x, \xi) \in \mathcal{U}\left(\left[-(D + \frac{\alpha}{2}), D + \frac{\alpha}{2}\right]\right) \times \mathcal{U}\left(\left[-(\frac{1}{D} + \frac{\beta}{2}), \frac{1}{D} + \frac{\beta}{2}\right]\right)$$

before rounding.

Example 3.32. *An example of values to be used could be $\alpha = 10^{-i}, \beta = 10^{-i}, D = 1$ for any $i \in \mathbb{N}_{>0}$. Increasing i would then increase the resolution of the lattice, and can be done for α and β individually.*

3.3.4 Higher Dimensional Case

The 1-dimensional case cannot readily be adapted to n dimensions via induction. One idea is instead to chose an α, β such that the lattice lies dense in Ω; it would then be possible to make a density argument and neglect rounding alltogether.

Chapter 4

Method

PyTorch [14] provides the required flexibility and features.

4.1 Setup

A uniform sample of the function to be approximated is created for a validation set. The training set consists of a high-resolution, equispaced sample of the function domain. The random seed is set for numpy, pytorch (including CUDA), and python. We create the architecture of our neural networks faithfully to Definition 2.3, adding skip connections when required.

4.2 Training

All weights are initialized pseudo-randomly under a uniform distribution to break symmetry in the gradient at creation time. We train every network using Stochastic Gradient Descent (SGD) on the Mean Squared Error Loss (MSE Loss) mean$((\Phi(x) - f(x))^2)$ via backpropagation. During experimentation the ADAM algorithm [15] provided similar results, however with slightly fewer iterations.

In the hopes of avoiding an abundance of "dead" ReLU's stuck with a 0-differential (and hence the network falling into a weak local minimum), we use Leaky ReLU's for training as inspired from the results of [16].

$$\text{lReLU}(x) := \begin{cases} x & \text{if } x > 0, \\ lx & \text{else} \end{cases}$$

for small $l = 0.01$. See the relevant experiment 5.1.

The learning rate is multiplied by a factor of 0.1 when a plateau in the validation error occurs. If the plateau remains, we break early, elsewise we break upon a fixed maximum number of epochs. We ensure a minimum amount of epochs. Everytime the network achieves a minimal validation error, we store a deep copy, this copy being the result of our training in case the error increases during the plateau.

If the weights of the networks are to be bounded, we set every individual parameter in the weights and biases of the affine transformations to $w = \max(-B, \min(B, w))$ after each step of the training algorithm.

4.3 Evaluation

Once training is done, lReLU's are swapped back for normal ReLU's. To evaluate our trained network, we create a larger test set of equispaced samples along the function domain to calculate $\|\cdot\|_{L^\infty}$ by the maximal difference.

Chapter 5

Experiments

We perform three experiments. We examine the effectiveness of Leaky ReLU training in our setting, its cause, and dead ReLU's as measure of a network. We are not able to reproduce the results in [16] for our purposes. Secondly we examine the ability of a neural network to approximate a basic function x^2 scaling in depth. Finally we apply the theory we have developed for uniformly sampling gabor systems.

5.1 Presence of "dead" ReLU's in relation to Leaky ReLU training

We define a dead ReLU as a ReLU which returns 0 for all the sampled input after it's application. We hypothesize that a large amount of such ReLU's cripple the network, and that LReLU's may help in preventing this from happening.

We examine the average presence of such dead ReLU's in a sample of 100 networks trained with the same architecture, in total and on a per layer basis. We compare the training results with and without Leaky ReLU training. We shall run the training algorithm in a simple problem space, defined by a network 16 wide and 4 deep, attempting to approximate $\cos(1x)$ on $(-2\pi, 2\pi)$. To contrast, we also run the training algorithm over a tougher problem space by incrasing the frequency $\cos(4x)$ and depth to 5. The remaining method is as described above 4.

Finally we corellate validation error with the number of dead ReLU's to examine whether it is a good indicator of a problem within our trained networks.

The results 5.1, 5.2 contradict our hypothesis. LReLU training did not seem to help significantly in reducing dead ReLU's. It is interesting to note the distribution of dead ReLU's, strongly skewed towards the last layer, scarce in the first. Also note that the harder to train network has many more dead ReLU's and a similar trend visible by layer.

Table 5.1: Percentage of Dead ReLU's, Simple (sample size = 100)

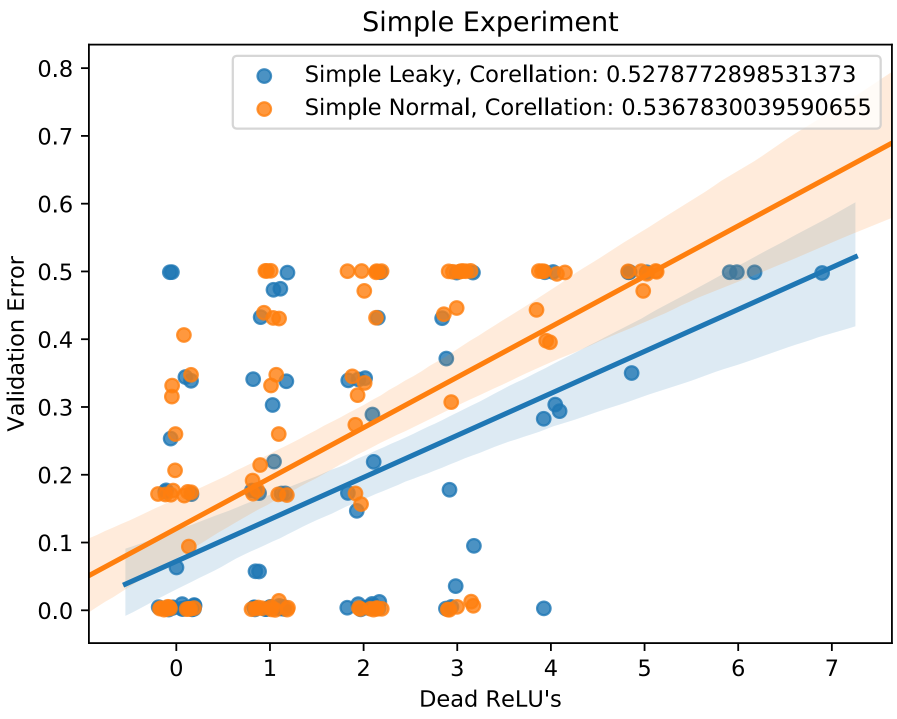
Layer	LReLU to Train	ReLU to Train
1	2.5%	2.25%
2	10.75%	9.25%
3	14.25%	14.25%
4	15.5%	17.75%
Total	10.75%	10.875%

Table 5.2: Percentage of Dead ReLU's, Difficult (sample size = 100)

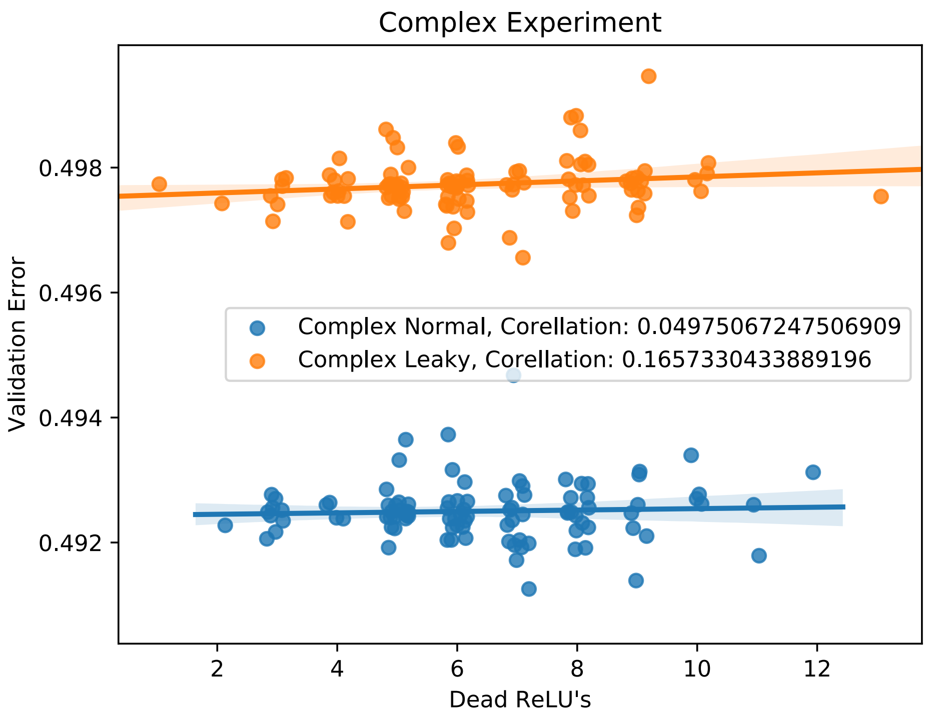
Layer	LReLU to Train	ReLU to Train
1	2.75%	3.75%
2	21.25%	21.75%
3	32.%	34.%
4	48.75%	48.5%
5	51.75%	52.%
Total	39.125%	40.0%

Further inspecting[1] the corellation between the number of dead ReLU's and the validation error (similar results are achieved for the testing error), we discover that indeed the measure of dead ReLU's seems to be a good indicator of a problem existing with the final result of the training algorithm. This can be explained by the network not using its full potential and restricting itself to a thinner, more shallow network in parts.

The corellation in the complex experiment for the leaky training method seems to be four times as high. It is possible that leaky training allows the network to make less use of its active ReLU's. The difference in ability between the two training methods is small.



[1]Note: the data points are slightly jittered in the graphs in order to display every sample, as they would strongly overlap otherwise.


Complex Experiment
Validation Error
0.498
0.496
0.494
0.492
Dead ReLU's
2
4
6
8
10
12
Complex Normal, Corellation: 0.04975067247506909
Complex Leaky, Corellation: 0.1657330433889196

5.2 Ability to approximate cos in depth and frequency

We would like to examine how the ability of neural networks to approximate a basic function such as x^2 changes with depth. We shall examine x^2 on $(0, 1)$. In order to achieve greater depths, we shall also make use of skip connections. The testing error shall be provided by the infinity norm to be consistent with the theory, this is calculated by taking 10^5 uniformly spaced samples of x^2 and the network.

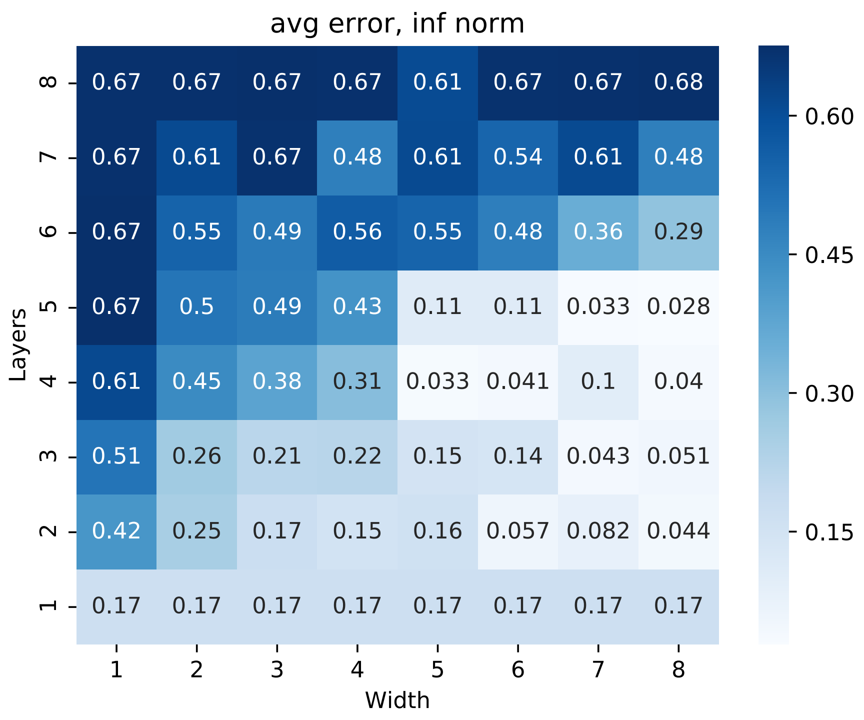


Figure 5.1: Heatmap of average (10 samples per data point) infinity norm error of networks approximating x^2 on $(0, 1)$. Method: normal training, no skip connections.

First results in Figure 5.1 and normal training can be explained well. When training depth beyond 5 we suffer strongly from the well-known vanishing gradients problem. As can be seen in Figure 5.2, the training algorithm is unable to train any network well and we end up with a constant function. At the opposite end of the spectrum, at depth 1 or 2, the network

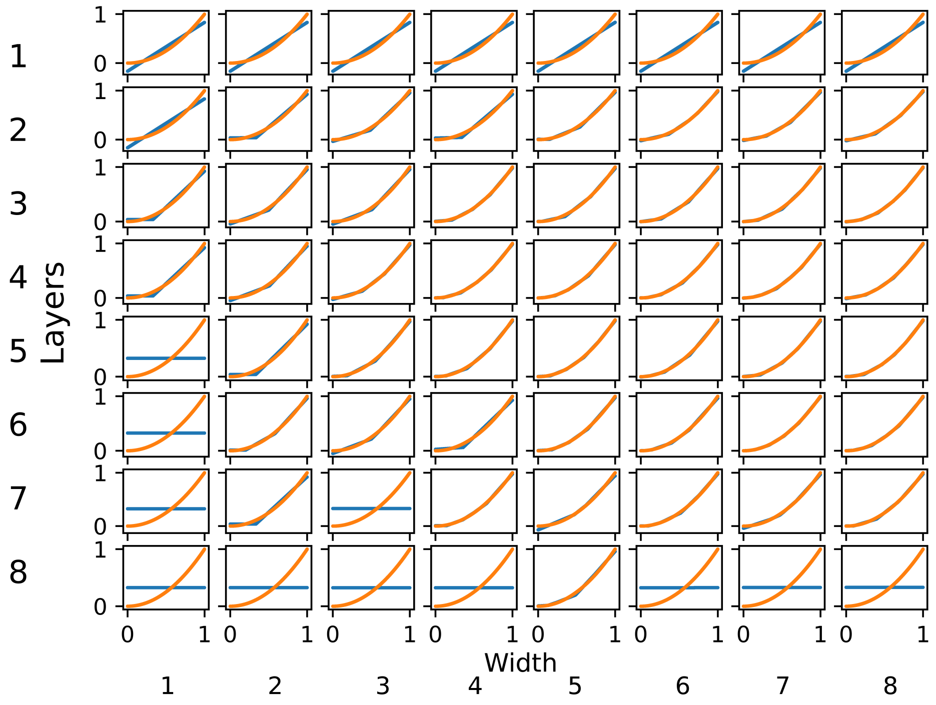


Figure 5.2: multiplot of the best network of sample of 10 (normal training, no skip connections)

has very little to work with, however is able to approximate decently with a single line.

Notice that for a fixed width, traveling down the layers, we notice a steady decrease in the error up to a certain point. After this begins a heavy dropoff as the algorithm fails by vanishing gradient. Interestingly, this "sweet spot" moves deeper as the network becomes wider. An explanation for this behaviour may be that wider algorithms are much easier to train. It also occurs very early and there is not enough data to discover a logarithmic trend.

Training with lReLu's (see figure 5.3) provides similar results, confirming our previous observations.

Finally, introducing skip connections, we see much stronger behaviour in depth. The vanishing gradient problem is stymied and at similar depths the networks perform much stronger.

We conclude that SGD does not suffice for training very deep networks. In order to take advantage of the theoretical strength of deeper networks,

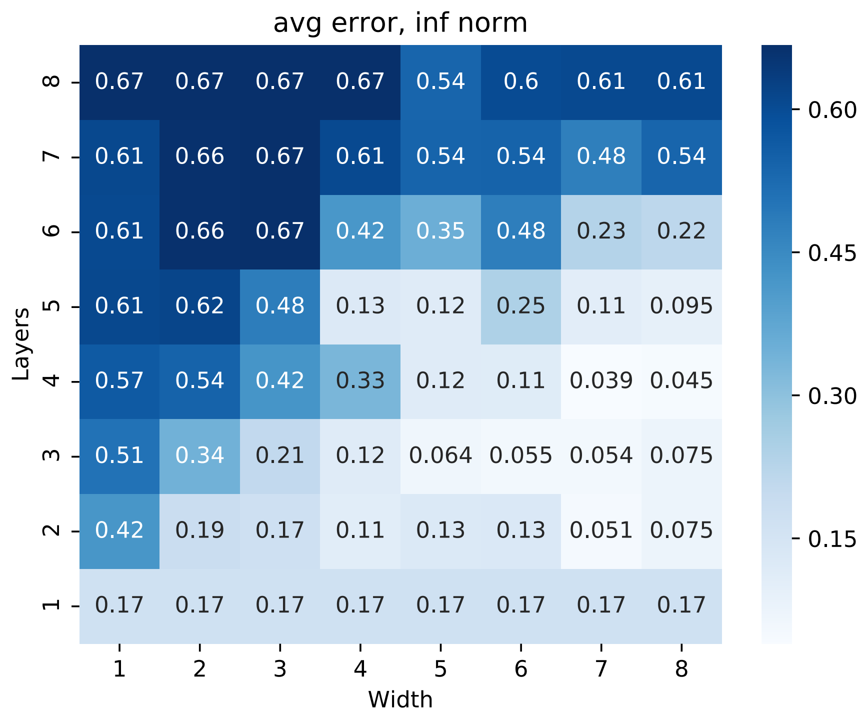
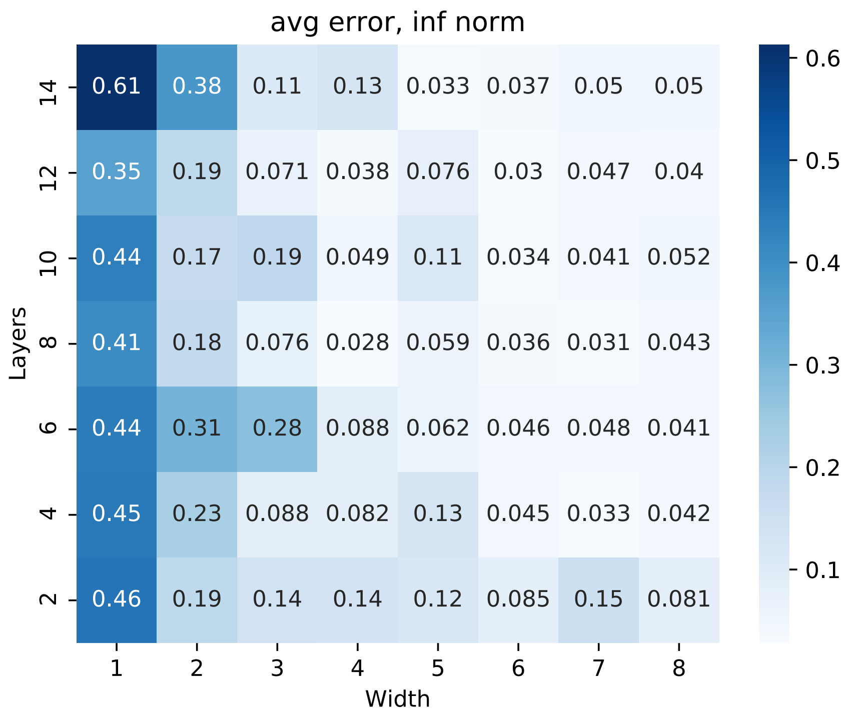


Figure 5.3: Heatmap of average (10 samples per data point) infinity norm error of networks approximating x^2 on $(0, 1)$. Method: LReLU training, no skip connections.

providing skip connections aids the training algorithm heavily. Providing sufficient width is also an important component in allowing the training algorithm to train deeper.



Layers	1	2	3	4	5	6	7	8
14	0.61	0.38	0.11	0.13	0.033	0.037	0.05	0.05
12	0.35	0.19	0.071	0.038	0.076	0.03	0.047	0.04
10	0.44	0.17	0.19	0.049	0.11	0.034	0.041	0.052
8	0.41	0.18	0.076	0.028	0.059	0.036	0.031	0.043
6	0.44	0.31	0.28	0.088	0.062	0.046	0.048	0.041
4	0.45	0.23	0.088	0.082	0.13	0.045	0.033	0.042
2	0.46	0.19	0.14	0.14	0.12	0.085	0.15	0.081

Figure 5.4: Heatmap of average (10 samples per data point) infinity norm error of networks approximating x^2 on $(0, 1)$. Method: skip connections and LReLU training.

5.3 Generalization in Gabor Systems

We examine how well a neural network of a given architecture can approximate the gabor system of a generator function g given it's ability to approximate g. We shall recall the theory we have developed for uniformly sampling gabor systems in 1 dimension. We shall make use of skip connections.

Note that the definition of our wilson base 3.27, with the generator g 3.29 can be demoted to a gabor system 3.21 by viewing the $\cos$ functions as the real part of the system, and the $\sin$ functions as the imaginary part. Since our theory is based on inspecting these components seperately with individual networks we do not sacrifice approximation quality. This demotion works with the paramaeters $\alpha = \frac{1}{2}, \beta = \frac{1}{4}$. We must also prepend g with a coefficient $\sqrt{2}g$ in order to give our gabor system precisely the ability to approximate the wilson base.

Further, in order to accomodate our theory for uniform sampling 3.3.3, we let $D = \frac{1}{4}$ and center $\sqrt{2}g$ around the origin $\sqrt{2}g(x - \frac{1}{4})$. Adapting and increasing the resolution of our lattice yields $\alpha = \frac{1}{4} \cdot 10^{-3}, \beta = \frac{1}{4} \cdot 10^{-3}$. We have arrived at a satisfying set of constants.

We uniformly sample $f_{i=1,2,\ldots,100} \in \mathcal{G}(\sqrt{2}g(x-\frac{1}{4}), \frac{1}{4}\cdot 10^{-3}, \frac{1}{4}\cdot 10^{-3}, [-\frac{1}{4}, \frac{1}{4}])$ and the base function $\sqrt{2}g(x - \frac{1}{4})$, with 10 data points each. We pick the Real or Imaginary component with a fair coin flip. We record the average $\| \cdot \|_{L^2}$ error 5.3.

Table 5.3: Average $\| \cdot \|_{L^2}$ Error, 6 Width (10 samples per point)

Depth	$\sqrt{2}g(x - \frac{1}{4})$	Avg. of 100 Uniform Samples f_i
4	0.000011231088163	0.001550457139979700
8	0.000023320751277	0.000741262240881042
12	0.000022420804288	0.000295524984503572

It is clear that g is much easier to approximate at every depth examined than the uniform sample points. Also clear is that the approximation of said points improves drastically as depth increases. Interestingly, depth 4 aproximates g better than depth $8, 12$; this may be because depth 4 suffices to approximate g very well and is much easier to train. It then becomes interesting that depth 8 and 12 do almost equally well.

We conclude that deep neural networks are not able to generalize from g to its translations and modulations very well, contrary to the theory, however that with depth and skip connections we are able to improve this ability drastically.

Bibliography

[1] P. Grohs, D. Perekrestenko, D. Elbraechter, and H. Boelcskei, "Deep neural network approximation theory," 2018, to be published.

[2] D. Perekrestenko, "Optimal approximation for gabor systems with relu neural networks," 2018, report - to be published.

[3] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *MCSS*, vol. 2, pp. 303–314, 1989.

[4] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251 – 257, 1991. [Online]. Available: http://www.sciencedirect.com/science/article/pii/089360809190009T

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[6] ——, "Identity mappings in deep residual networks," *CoRR*, vol. abs/1603.05027, 2016. [Online]. Available: http://arxiv.org/abs/1603.05027

[7] D. Yarotsky, "Error bounds for approximations with deep relu networks," *CoRR*, vol. abs/1610.01145, 2016. [Online]. Available: http://arxiv.org/abs/1610.01145

[8] E. Ott, *Chaos in Dynamical Systems*, 2nd ed. Cambridge University Press, 2002.

[9] P. Grohs, *Optimally Sparse Data Representations*. Cham: Springer International Publishing, 2015, pp. 199–248. [Online]. Available: https://doi.org/10.1007/978-3-319-18863-8_5

[10] D. L. Donoho, "Unconditional bases are optimal bases for data compression and for statistical estimation," *Applied and Computational Harmonic Analysis*, vol. 1, no. 1, pp. 100 – 115, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1063520383710080

[11] P. Auscher, "Remarks on the local fourier bases," pp. 203–218, 01 2019.

[12] P. Auscher, G. Weiss, and M. Wickerhauser, "Local sine and cosine bases of coifman and meyer and the construction of smooth wavelets," 05 1999.

[13] M. E. Muller, "A note on a method for generating points uniformly on n-dimensional spheres," *Commun. ACM*, vol. 2, no. 4, pp. 19–20, Apr. 1959. [Online]. Available: http://doi.acm.org/10.1145/377939.377946

[14] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[16] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *CoRR*, vol. abs/1505.00853, 2015. [Online]. Available: http://arxiv.org/abs/1505.00853

Appendix A

Code

For the code, see https://github.com/ZirconCode/DNN-approx-public.