# The Expressive Power of Neural Networks: A Trade-off Between Depth and Width

CS639 Final Project Report

**Shengli Jiang**

**Yanpeng Zhao**

University of Wisconsin-Madison

December 9, 2020

# 1 Introduction

In the past decade, deep learning has proven to be a very powerful tool for solving a wide range of problems, from image recognition to signal prediction. Despite their success, many of the techniques used in artificial neural networks are heuristics with no theoretical guarantees. A key factor to mathematically analyze the neural network is its expressive power (i.e. the ability to approximate functions). In the views of depth and width, people have extensively studied the expressive power of neural networks. Telgarsky showed that standard types of networks always gain in expressive power with the addition of layers [1, 2]. Since people examine the expressive power from various angles and with distinct methods (e.g. approximate univariate polynomials and multivariate polynomials), it is necessary to systematically generalize current methods and results.

The report is organized as follows:

- Section 2 introduces the concept of neural network.
- Section 3 discusses the definition of expressive power and universal approximation theorem.
- Section 4 covers some popular methods that measure the expressive power.
- Section 5 focuses on using oscillation method to show the efficiency of network depth.
- Section 6 mentions another view of analyzing the efficiency of network width.

# 2 Neural Networks

A neural network is a computation model defined by an acyclic directed graph. A node waits for values on its incoming edges and computes functions of these values that are transmitted along its outgoing edges. Different nodes can compute different functions, such as Rectified Linear Unit (ReLU) and tanh. A neural network has a root node that receives and computes the input vector. The intermediate nodes apply their computation and output to other nodes. A characteristic of neural network is its layer structure. The computation result of one layer is passed to the next layer; with more layers, the associated function of a neural network can be highly nonlinear and mostly nonconvex. A sample neural network architecture is shown in Figure 1.

**Definition 2.1** ([3] Neural network)**.** A network $A$ is defined by a directed acyclic graph $G_A$, an activation function $\sigma$, and a set of parameters of one weight for each edge and one bias for each node of $G_A$. Then, if the longest path in the graph as length $n+1$, we say that the network has $n$ layers. Then, the input layer (layer 0) is defined to be the set of nodes with in-degree 0. To create a function from $\mathbb{R}^d \mapsto \mathbb{R}$, we assume that there are $d$ such input nodes and there is only one output node, the unique node in layer $n$. Then for $1 \leq i \leq n$ we define layer $i$ to be the set of nodes with a predecessor in layer $i-1$ and no predecessor in any layer $j \geq i$. Then, computation of the function $f_A$ proceeds by computing the output of each subsequent layer $i$. For each node $u$ in layer $i$, take vector of outputs of units $v$ with directed edges to $u$ along with the vector $w$ of weights associated to those edges and the bias $b$ at node $u$. Then $u$ outputs $\sigma\left(w^\intercal x + b\right)$. The only exception is that the output node in layer $n$ does not apply $\sigma$ and rather outputs $w^\intercal x + b$.

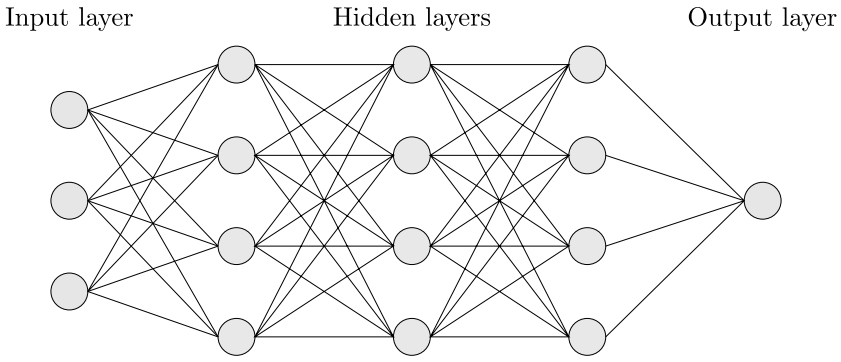


Figure 1: A neural architecture structure with an input vector of $\mathbb{R}^3$ and an output scalar $\mathbb{R}$. The lines between computation nodes represent the weights.

In this report, we focus on the neural networks with an input $x \in \mathbb{R}^m$ and a binary output $\{-1, 1\}$. We denote a fully connected neural network with $n$ hidden layers of width $k$ as $A_{(n,k)}$. We also define the associated function of architecture $A_{(n,k)}$ as $F_{A_{(n,k)}}(x;\theta)$, where $x$ is an input and $\theta$ represents all the parameters in the network.

Since most neural networks now use ReLU as their activation function, we define the ReLU units as

$$\begin{aligned} v_{out} &= \sigma\left(Av_{in} + b\right), \\ \sigma(z) &= \max\left(0, z\right), \end{aligned} \tag{1}$$

where $v_{in} \in \mathbb{R}^m$ and $v_{out} \in \mathbb{R}^n$ are the input and output vectors, respectively. $A \in \mathbb{R}^{n\times m}$ is the weight matrix. $b \in \mathbb{R}^n$ is a bias vector that adds flexibility for learning. In the following study, we focus on the use of ReLU units to convey the idea of expressive power.

## 3 Expressive Power of Neural Networks

The expressive power describes neural networks' ability to approximate functions. Before delving into expressive power, we need to introduce the Universal Approximation Theorem, which is the result of a series of papers from Cybenko [4], Hornik [5], Leshno [6] and Pinkus [7]. In the following analysis, we will cover some theorems in functional analysis, the proofs of which can be found in [8]. For the concepts of measure theory, people can refer to [9]. We have adopted and adjusted some of the analysis in [10].

**Theorem 3.1** (Geometric Hahn-Banach Theorem)**.** *Let $V$ be a normed vector space and $A, B \subset V$ be two non-empty, closed, disjoint and convex subsets such that one of them is compact. Then there exist a continuous linear function $f \not\equiv 0$, some $\alpha \in \mathbb{R}$ and an $\epsilon > 0$ such that $f(x) \leq \alpha - \epsilon$ for any $x \in A$ and $f(y) \geq \alpha + \epsilon$ for any $y \in B$.*

**Corollary 3.1.1.** *Let $V$ be a normed vector space over $\mathbb{R}$ and $U \subset V$ be a linear subspace such that $cl(U) \neq V$, where $cl(U)$ is a closure of the subset $U$. Then there exists a continuous linear map $f : V \mapsto \mathbb{R}$ with $f(x) = 0$ for any $x \in U$, and $f \not\equiv 0$.*

**Theorem 3.2** (Lebesgue Bounded Convergence Theorem)**.** *Let $X$ be a measure space, $\mu$ be a Borel measure on $X$, $g : X \mapsto \mathbb{R}$ be a $L^1$ function, and $\{f_n\}$ be a sequence of measurable functions from*

*$X \mapsto \mathbb{R}$ such that $|f_n(x)| \leq g(x)$ for all $x \in X$ and $\{f_n\}$ converges point-wise to a function $f$. The $f$ is integrable and*

$$\lim_{n\to\infty} \int f_n(x)\, d\mu(x) = \int f(x)\, d\mu(x). \tag{2}$$

**Theorem 3.3** (Riesz Representation Theorem)**.** *Let $\Omega$ be a subset of $\mathbb{R}^n$ and $F : C(\Omega) \mapsto \mathbb{R}$ be a linear functional on the space of continuous real functions with domain on $\Omega$. Then there exists a signed Borel measure $\mu$ on $\Omega$ such that for any $f \in C(\Omega)$, we have that*

$$F(f) = \int_\Omega f(x)\, d\mu(x). \tag{3}$$

**Definition 3.1.** Let $I_n$ denote the $n$-dimensional unit cube, $[0,1]^n$. The space of continuous functions on $I_n$ is denoted by $C(I_n)$. For $\sigma : \mathbb{R} \mapsto \mathbb{R}$ an activation function, set

$$\Sigma_n(\sigma) = \text{span}\{\sigma(y^\intercal x + \theta) \,|y \in \mathbb{R}^n, \theta \in \mathbb{R}\}. \tag{4}$$

The set $\Sigma_n(\sigma)$ consists of all the functions that can be calculated by a neural network with a single hidden layer and activation function $\sigma$.

**Definition 3.2.** We say that a neural network with activation function $\sigma : \mathbb{R} \mapsto \mathbb{R}$ is a universal approximator on $I_n$ if $\Sigma_n(\sigma)$ is dense in $C(I_n)$. $\Sigma_n(\sigma)$ is dense in $C(I_n)$ if every point in $C(I_n)$ either belongs to $\Sigma_n(\sigma)$ or is a limit point of $\Sigma_n(\sigma)$; that is, the closure of $\Sigma_n(\sigma)$ is constituting the whole set $C(I_n)$.

**Definition 3.3.** Let $n \in \mathbb{N}_1$. We say an activation function $\sigma : \mathbb{R} \mapsto \mathbb{R}$ is $n$-discriminatory if the only signed Borel measure $\mu$ such that

$$\int_{I_n} \sigma(y^\intercal x + \theta)\, d\mu(x) = 0 \qquad \forall\ y \in \mathbb{R}^n, \text{ and } \theta \in \mathbb{R} \tag{5}$$

is the zero measure.

**Definition 3.4.** We say that $\sigma$ is sigmoidal if

$$\sigma(x) \to \begin{cases} 1 & \text{as } x \to +\infty, \\ 0 & \text{as } x \to -\infty. \end{cases} \tag{6}$$

Based on Cybenko's proof, to prove that a neural network with a single layer of ReLU units is a universal approximator, we will first show that the ReLU function is a discriminatory function. Then we will show that a neural network with continuous discriminatory functions as activation functions is a universal approximator.

**Lemma 3.4.** *Any continuous sigmoidal function $\sigma$ is discriminatory.*

*Proof.* For any, $x, y, \theta, \phi$ we have

$$\sigma(\lambda(y^\intercal x + \theta) + \phi) \begin{cases} \to 1 & \text{for } y^\intercal x + \theta > 0 \text{ as } \lambda \to +\infty, \\ \to 0 & \text{for } y^\intercal x + \theta < 0 \text{ as } \lambda \to +\infty, \\ = \sigma(\phi) & \text{for } y^\intercal x + \theta = 0 \text{ for all } \lambda. \end{cases} \tag{7}$$

Thus, we can see the function $\sigma_\gamma(x)$ converges to the function

$$\gamma(x) = \begin{cases} 1 & \text{for } y^\intercal x + \theta > 0 \\ 0 & \text{for } y^\intercal x + \theta < 0 \\ \sigma(\phi) & \text{for } y^\intercal x + \theta = 0, \end{cases} \tag{8}$$

as $\lambda \to +\infty$.

Let $\Pi_{y,\theta}$ be the hyperplane defined by $\{x | y^\intercal x + \theta = 0\}$ and $H_{y,\theta}$ be the open half-space defined by $\{x | y^\intercal x + \theta > 0\}$. Then by the Lebesgue Bounded Convergence Theorem and Definition 3.3, we have that

$$\begin{aligned} 0 &= \int_{I_n} \sigma_\gamma(x)\, d\mu(x) \\ &= \int_{I_n} \gamma(x)\, d\mu(x) \\ &= \sigma(\phi)\, \mu(\Pi_{y,\theta}) + \mu(H_{y,\theta}) \end{aligned} \tag{9}$$

for all $\phi, \theta, y$.

We now show that the measure of all half-planes being 0 implies that the measure $\mu$ must be 0. Fix $y$. For a bounded measurable function, $h$, define the linear functional, $F$, according to

$$F(h) = \int_{I_n} h(y^\intercal x)\, d\mu(x) \tag{10}$$

and note that $F$ is a bounded functional on $L^\infty(\mathbb{R})$ since $\mu$ is a finite signed measure. Let $h$ be the indicator function of the interval $[\theta, \infty)$ (i.e. $h(u) = 1$ if $u \geq \theta$ and $h(u) = 0$ if $u < \theta$) so that

$$F(h) = \int_{I_n} h(y^\intercal x)\, d\mu(x) = \mu(\Pi_{y,-\theta}) + \mu(H_{y,-\theta}) = 0. \tag{11}$$

Similarly, $F(h) = 0$ if $h$ is the indicator function of the open interval $(\theta, \infty)$. By linearity, $F(h)$ for the indicator function of any interval and hence for any simple function (i.e. sum of indicator functions of intervals). Since simple functions are dense in $L^\infty(\mathbb{R})$, $F = 0$. Additionally, since sin and cos are elements of $L^\infty(\mathbb{R})$, we can use that

$$\begin{aligned} F(\cos) + iF(\sin) &= \int_{I_n} (\cos(y^\intercal x) + i \sin(y^\intercal x))\, d\mu(x) \\ &= \int_{I_n} e^{iy^\intercal x} d\mu(x) \\ &= 0. \end{aligned} \tag{12}$$

This is true for all $y \in \mathbb{R}^n$, which means that the Fourier transform of $\mu$ is 0. This is only possible if $\mu = 0$, and we are done with the proof. □

**Lemma 3.5.** *The ReLU function is 1-discriminatory.*

*Proof.* Let $\mu$ be a signed Borel measure, and assume the following holds for all $y \in \mathbb{R}$ and $\theta \in \mathbb{R}$:

$$\int \text{ReLU}(yx + \theta)\, d\mu x = 0. \tag{13}$$

We want to show $\mu = 0$. Since ReLU is unbounded, we can construct a bounded continuous sigmoidal function from subtracting two ReLU functions with different parameters. Specifically, consider the function

$$\sigma(x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{if } 0 \leq x \leq 1, \\ 1 & \text{if } x > 1. \end{cases} \tag{14}$$

Then any function of the form $g(x) = \sigma(yx + \theta)$ with $y \neq 0$ can be defined as

$$g(x) = \text{ReLU}(yx + \theta_1) - \text{ReLU}(yx + \theta_2), \tag{15}$$

, where $\theta_1 = -\theta/y$, $\theta_2 = (1 - \theta)/y$, and for all $y \in \mathbb{R}$ and $\theta \in \mathbb{R}$. If $y = 0$, then

$$g(x) = \sigma(\theta) = \begin{cases} \text{ReLU}(\sigma(\theta)) & \text{if } \sigma(\theta) \geq 0, \\ -\text{ReLU}(\sigma(-\theta)) & \text{if } \sigma(\theta) < 0. \end{cases} \tag{16}$$

Thus, for any $y \in \mathbb{R}$, $\theta \in \mathbb{R}$,

$$\begin{aligned} \int \sigma(yx + \theta)\, d\mu(x) &= \int (\text{ReLU}(yx + \theta_1) - \text{ReLU}(yx + \theta_2))\, d\mu(x) \\ &= \int \text{ReLU}(yx + \theta_1)\, d\mu(x) - \int \text{ReLU}(yx + \theta_2)\, d\mu(x) \\ &= 0. \end{aligned} \tag{17}$$

Only both integrals over each ReLU must have measure zero if integral of $\sigma$ has measure zero. Therefore, ReLU is 1-discriminatory. □

**Lemma 3.6.** *If $\Sigma_1(\sigma)$ is dense in $C(I)$ then $\Sigma_n$ is dense in $C(I_n)$.*

*Proof.* We use the fact that the span of the set $\{f(z^\intercal x) \,|\, y \in \mathbb{R}^n, g \in C(I)\}$ is dense in $C(I_n)$. That is, given any function $g \in C(I_n)$ and $\epsilon > 0$, there exists function $f_i$ in $C(I)$ such that

$$\left| g(x) - \sum_{i=1}^{N} f_i(z_i^\intercal x) \right| < \frac{\epsilon}{2}. \tag{18}$$

If we now examine each function $f_i(z_i^\intercal x)$ and use the assumption that $\Sigma_1(\sigma)$ is dense in $C(I)$, we conclude that for any such function, there exists a sum of functions such that

$$\left| f_i(z_i^\intercal x) - \sum_{j=1}^{N_i} \sigma\left(y_{i,j}^\intercal x + \theta_{i,j}\right) \right| < \frac{\epsilon}{2i}. \tag{19}$$

By applying the triangle inequality, we get that

$$\begin{aligned} \left| g(x) - \sum_{i=1}^{N} \sum_{j=1}^{N_i} \sigma\left(y_{i,j}^\intercal x + \theta_{i,j}\right) \right| &< \left| g(x) - \sum_{i=1}^{N} f_i(z_i^\intercal x) \right| + \frac{k\epsilon}{2k} \\ &< \frac{\epsilon}{2} + \frac{\epsilon}{2} \\ &= \epsilon. \end{aligned} \tag{20}$$

This shows we can get arbitrarily close to any function in $C(I_n)$ by using functions in $\Sigma_n(\sigma)$. $\square$

**Theorem 3.7.** *Let $\sigma$ be a continuous discriminatory function. Then a neural network with $\sigma$ as the activation function is a universal approximator.*

*Proof.* For the sake of contradiction, assume $\Sigma_n(\sigma)$ is not dense in $C(I_n)$. Let $cl(S)$ be the closure of $S$, and it is a closed proper subspace of $C(I_n)$. By the Hahn-Banach theorem, there is a bounded linear function on $C(I_n)$, call it $L$, with the property that $L \neq 0$ but $L(cl(S)) = L(\Sigma_n(\sigma)) = 0$. By the Riesz Representation Theorem, there exists some Borel measure $\mu$ such that

$$L(g) = \int_{I_n} g(x)\, d\mu(x) \qquad \forall g \in C(I_n). \tag{21}$$

However, since for any $y$ and $\theta$ the function $\sigma(y^\intercal x + \theta)$ is an element of $cl(S)$, this means that for all $y \in \mathbb{R}^n$, and $\theta \in \mathbb{R}$, we have $\int_{I_n} \sigma(y^\intercal x + \theta)\, d\mu(x) = 0$, which means that $\mu = 0$ (since $\sigma$ is discriminatory) and therefore $L(g) = 0$ for any $g \in C(I_n)$. This contradicts the corollary of the Hahn-Banach Theorem, and thus finishes the proof. $\square$

As a direct consequence of Theorem 3.7 and previous Lemmas, we arrive at the desired result:

**Corollary 3.7.1** (Universal Approximation Theorem)**.** *Neural networks with ReLU activation functions are universal approximators.*

This theorem implies that a single layer neural network with arbitrarily many ReLU units can be used to approximate arbitrarily complex functions with arbitrarily precision. However, it does not tell us how to choose the neural network parameters to achieve the function that we want to approximate.

# 4 Methods to Measure Expressive Power

Many creative approaches have been proposed to measure the expressive power of neural networks. In this section, we will briefly describe some of the popular approaches. We will not provide detailed proof for each theorem in this section. In Section 5, we will discuss detailed proofs for one of the methods.

**Trajectory Length** The notion of of trajectory and trajectory length is proposed by Raghu et al. [11]. Let the ReLU-unit network have an architecture $A$, and its associated function be $F_A(x;\theta)$, where $x \in \mathbb{R}^m$ is the input, and $\theta$ is the weight vector. Precisely quantifying the properties of $F_A(x;\theta)$ over the entire input space is intractable. Thus, Raghu et al. defined *trajectories* as follows

**Definition 4.1.** Given two points, $x_0, x_1 \in \mathbb{R}^m$, $x(t)$ is a trajectory between $x_0$ and $x_1$ if $x(t)$ is a curve parameterized by a scalar $t \in [0,1]$, with $x(0) = x_0$ and $x(1) = x_1$.

Another related quantity, *trajectory length* is defined as follows

**Definition 4.2.** Given a trajectory $x(t)$, its length $l(x(t))$ is the standard arc length:

$$l(x(t)) = \int_t \left|\left| \frac{dx(t)}{dt} \right|\right| dt. \tag{22}$$

**Theorem 4.1** ([11] Theorem 3)**.** *Bound on Growth of Trajectory Length. Consider a ReLU network $A_{n,k}$, where the depth is $n$ and width is $k$. Define $z^{(d)}(x(t)) = z^{(d)}(t)$ to be the image of the trajectory in layer $d$ of the network, we have*

$$\mathbb{E}\left[l\left(z^{(d)}(t)\right)\right] \geq O\left(\frac{\sigma\sqrt{k}}{\sqrt{k+1}}\right)^d l(x(t)) \tag{23}$$

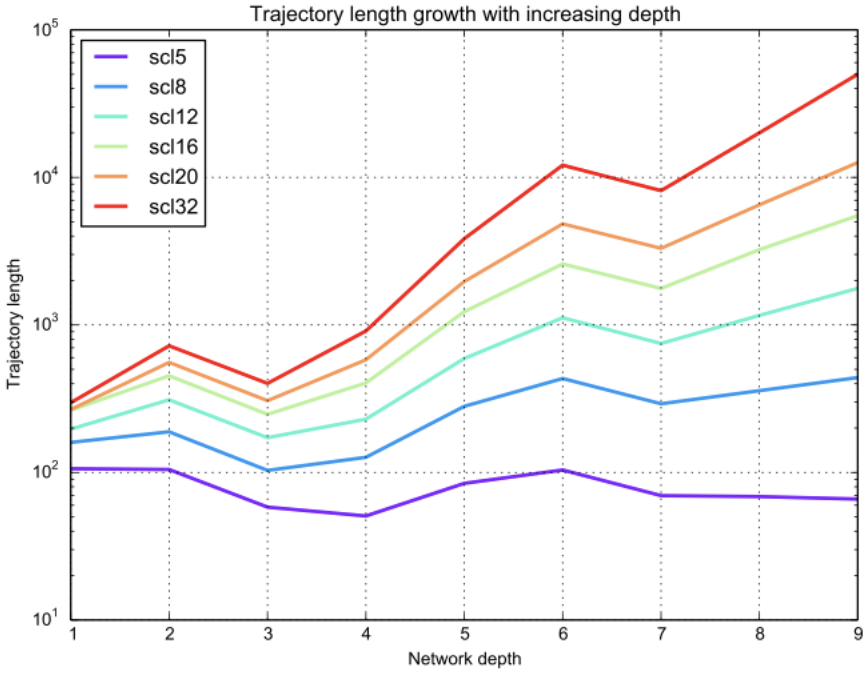


Figure 2: Trajectory growth with different initialization scales. The decrease in trajectory length in layers 3 and 7 is due to the fact that they are pooling layers. Figure is from [11].

It is shown that the trajectory length $l(x(t))$ grows exponentially with depth $d$. A schematic drawing of the relationship between trajectory length and network depth is shown in Figure 2. In summary, they have shown that the complexity of the computed function grows exponentially with depth.

**Oscillation** Telgarsky constructed a set of uniformly spaced alternating labels in the unit interval [1, 2]. A function that hits all the points in a set must oscillate very quickly. His idea was to show that shallow networks cannot accomplish such oscillations, while deep networks can. We discuss his approach in detail in Section 5.

**Counting Linear Regions** A network with a piece-wise linear activation function, such as ReLU, defines a piece-wise linear function. A measure of the complexity of a piece-wise linear function is the number of linear pieces required to describe the function. A network architecture has more expressive power if it can represent a function in terms of more linear regions. Montufar et al. [12] has given a lower bound on the maximal number of linear regions defined by a feed-forward network of given dimensions.

**Theorem 4.2** ([12], Theorem 4)**.** *Let $A$ be a network of depth $n$ and layer width $k_i > d$ for $1 \leq i \leq n$ defining a function $F_A : \mathbb{R}^d \mapsto \mathbb{R}$. Then maximal number of linear regions defined by $F_A$ is at least*

$$\left(\prod_{i=1}^{n-1} \left\lfloor \frac{k_i}{d} \right\rfloor^d\right) \sum_{j=0}^{d} \binom{k_n}{j}. \tag{24}$$

**VC Dimension** Another way to measure the network expressive power is to measure the VC dimension. Bartlett et al. has proven some nearly tight bounds on the VC dimensions of neural networks based on depth [13].

**Theorem 4.3** ([13], Theorems 3)**.** *The class $\mathcal{F}_{k,n}$ of networks with piece-wise linear activation functions of width $k$, depth $n$ and size $|A| = k^2 n$ has VC dimension bounded by:*

$$\Omega\left(|A| n \log\left(|A|/n\right)\right) = \Omega\left(k^2 n^2 \log\left(k^2\right)\right). \tag{25}$$

# 5 The Efficiency of Network Depth

In order to demonstrate the efficiency of network depth, it is common to show that some of the functions that can be represented by a deep architecture require a shallow network that is exponentially larger than the size of the deep network. These results were originally demonstrated by Telgarsky in [1] in a constrained setting and were generalized in [2]. Some of the following results are adopted and adjusted from [14].

Telgarsky used classification error as the measure of approximation capability in [1]. Recall the definition of an associated function of a neural network architecture $A_{(n,k)}$ is $F_{A_{(n,k)}}(x;\theta)$, which has a depth $n$ and a width $k$. The input is $x$ and the weight vector is $\theta$. Given a function $f : \mathbb{R} \mapsto \mathbb{R}$, let $\tilde{f} : \mathbb{R} \mapsto \{0,1\}$ denote the corresponding classifier $\tilde{f}(x) := \mathbb{1}[f(x) \geq 1/2]$. Given a sequence of points $(x_i, y_i) \in \mathbb{R} \times \{0,1\}$ for $1 \leq i \leq m$, the classification error $\mathcal{R}$ is defined as $\mathcal{R}(f) := \frac{1}{m} \sum_i \mathbb{1}[\tilde{f}(x_i) \neq y_i]$.

**Definition 5.1.** We say that $\sigma : \mathbb{R} \mapsto \mathbb{R}$ is *t-sawtooth* if it is piece-wise affine with $t$ pieces, meaning $\mathbb{R}$ is partitioned into $t$ consecutive intervals, and $\sigma$ is affine within each interval. For example, ReLU is 2-sawtooth.

**Definition 5.2.** Let $F_{A_{(n,k;t)}}$ denote $t$ iterations of a recurrent network with a depth $n$ and width $k$. Every $f \in F_{A_{(n,k;t)}}$ consists of some fixed network $g \in F_{A_{(n,k)}}$ applied $t$ times:

$$f(x) = g^k(x) = \underbrace{(g \circ \cdots \circ g)}_{t \text{ times}}(x). \tag{26}$$

Consequently, $F_{A_{(n,k;t)}} \subseteq F_{A_{(nt,k)}}$, but the former has $\mathcal{O}(nk)$ parameters whereas the latter has $\mathcal{O}(ntk)$ parameters.

**Theorem 5.1** ([1], Theorem 1.1). *There exists a network $A_{(2h,2)}$ (i.e. depth $2h$ and width $2$) that computes a function $F_A : \mathbb{R} \mapsto \mathbb{R}$ and a set $S$ of $m := 2^h$ points in $[0,1] \times \{0,1\}$ such that $\mathcal{R}(F_A) = 0$ and any network $B$ of depth $l$ and width $w$ such that $w \leq 2^{(h-3)/l-1}$ has $\mathcal{R}(F_B) \geq 1/6$.*

To prove this result, we first need to define a set $S = \{\left(\frac{i}{m}, \mathbb{1}[i \text{ is odd}]\right) : 0 \leq i \leq m\}$. $S$ is a set of uniformly spaced alternating labels within a unit interval. A function that hits all the points in $S$ must oscillate very quickly. The core idea in [1] is to show that shallow networks cannot accomplish such oscillations and to provide a deep network that can. The proof of Theorem 5.1 is based on the following lemmas.

**Lemma 5.2** ([1], Lemma 2.3). *Let $f, g : \mathbb{R} \mapsto \mathbb{R}$ be respectively $t_f$- and $t_g$-sawtooth. Then $f + g$ is $(t_f + t_g)$-sawtooth, and $f \circ g$ is $t_f t_g$-sawtooth.*

*Proof.* Let $\mathcal{I}_f, \mathcal{I}_g$ be partitions of $\mathbb{R}$ corresponding to pieces of the respective functions. Now for any intervals $U_f \in \mathcal{I}_f$ and $U_g \in \mathcal{I}_g$, the function $f + g$ has a single slope on the interval $U_f \cap U_g$. So, $f + g$ is $|U_f \cap U_g|$-sawtooth, where the intersection of the partition is defined as the set of all piece-wise intersections. By sorting all intervals of $\mathcal{I}_f, \mathcal{I}_g$ by their left endpoint, with each endpoint defining at most one interval intersection, the number of pieces is $|U_f \cap U_g| < t_f + t_g$.

Now take an interval $U_g \in \mathcal{I}_g$. Then $f \circ g(U_g)$ contains at most $|\mathcal{I}_f| = t_f$ pieces because $g$ is a single-slope affine on $U_g$ so $g(U_g)$ is a continuous interval. Since $U_g$ is arbitrary, we have $f(g)$ which is $t_f t_g$-sawtooth. □

**Lemma 5.3** ([1], Lemma 2.1). *If an activation function $\sigma$ is $t$-sawtooth, then a network $B_{(l,w)}$ calculates a function $F_B$ that is $(tw)^l$-sawtooth.*

*Proof.* We can induct over the layers to show that $F_B$ is $(tw)^l$-sawtooth. Specifically, the induction will show that the output of each node in layer layer $i$ is $(tw)^i$-sawtooth as a function of the input $x$. For the first layer, each node calculates $x \mapsto \sigma(a^\intercal x + b)$ for some weight vector $a$ and bias $b$. This function is $tw$-sawtooth since the only non-linearity is induced by the activation function. Assume inductively that each node at layer $i-1$ calculates a piece-wise linear function $g_j$ for $1 \leq j \leq w$ that is $(tw)^{i-1}$-sawtooth. Then the function at a node in layer $i$ is $x \mapsto \sigma\left(\sum_j a_j g_j(x) + b\right)$ for some new weights $a$ and bias $b$. By the above statements, this function is $tw(tw)^{i-1} = (tw)^i$-sawtooth. □

**Lemma 5.4** ([1], Lemma 2.2). *Take $S = \{\left(\frac{i}{m}, \mathbb{1}[i \text{ is odd}]\right) : 0 \leq i \leq m\}$, as defined above. Then every $t$-sawtooth $f : \mathbb{R} \mapsto \mathbb{R}$, satisfies $\mathcal{R}(f) \geq \frac{m-4t}{3m}$.*

*Proof.* Since $f$ is $t$-sawtooth, it is piece-wise monotonic on each of the $t$-piece partition of $\mathbb{R}$, so $f$ can traverse $1/2$ at most $2t-1$ times (i.e. 1 crossing for each interval, and 1 crossing for each right endpoint of an interval, except for the last one). Therefore, $\tilde{f}$ is piece-wise constant with at most $2t$ piece. Then to put $m$ points into $2t$ intervals, at least $m - 4t$ points must fall into intervals with at least 3 points. Since $S$ contains labels with alternating 0 and 1, any interval with at least one-third of points (i.e. $\frac{m-4t}{3}$) will be misclassified, so $\mathcal{R}(f) \geq \frac{m-4t}{3m}$. □

**Lemma 5.5** ([1], Lemma 2.4). *Let $S_j = \{\left(\frac{i}{2^j}, \mathbb{1}[i \text{ is odd}]\right) : 0 \leq i < 2^j\}$. Then we can construct a* ReLU *network $A_{(2j,2)}$ of width 2 and depth $2j$ such that $F_{A_{(2j,2)}}(x_i) = y_i$ for all $(x_i, y_i) \in S_j$.*

*Proof.* Consider the *mirror map* $f_m : \mathbb{R} \mapsto \mathbb{R}$:

$$f_m(x) := \begin{cases} 2x & 0 \le x \le 1/2 \\ 2(1-x) & 1/2 < x \le 1 \\ 0 & \text{otherwise.} \end{cases} \tag{27}$$

The mirror map examples are shown in Figure 3. Note that $f_m \in F_{A_{(2,2)}}$. Let $\sigma$ be the ReLU function. Then $f_m(x) = \sigma(2\sigma(x) - 4\sigma(x - 1/2))$ can easily be defined a neural network of two hidden layers with width 2.

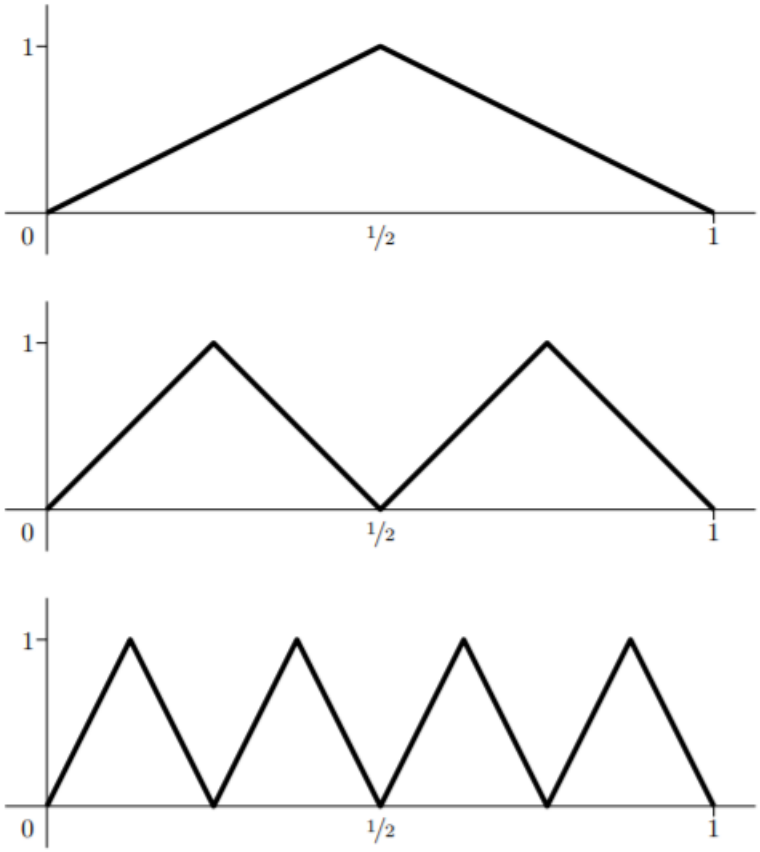


Figure 3: $f_m, f_m^2, f_m^3$ [1].

Now we want to induct over $j$ to prove the following claim. For each $x \in [0,1]$ choose $i_j \in \{0, 1, \cdots, 2^{j-1}\}$ and $x_j \in [0,1)$ such that $x = (i_j + x_j)\, 2^{1-j}$. Then we claim

$$f_m^j(x) = \begin{cases} 2x_j & 0 \le x_j \le 1/2 \\ 2(1-x_j) & 1/2 < x_j < 1, \end{cases} \tag{28}$$

where the definition of $f_m^j$ can be seen from Definition 5.2.

When $j = 1$, it was proven by the definition of $f_m$. For the inductive step, assume this holds for $j$ we want to show it for $j+1$. Now note that for any function $g : \mathbb{R} \mapsto \mathbb{R}$, pre-composition $g \circ f_m$

calculates

$$g \circ f_m = \begin{cases} g(2x) & 0 \leq x \leq 1/2 \\ g(2-2x) & 1/2 < x \leq 1. \end{cases} \tag{29}$$

In other words $g \circ f_m$ scales $g$ horizontally by a half and reflects it about the vertical line at $x = 1/2$. Using this reflection property and then the symmetry of $f_m^j$ from the inductive hypothesis, we can note that for $x \in [0, 1/2]$:

$$\left(f_m^j \circ f_m\right)(x) = \left(f_m^j \circ f_m\right)(1-x) = \left(f_m^j \circ f_m\right)(x+1/2). \tag{30}$$

For $x \in [0, 1/2]$, w.l.o.g we obtain $\left(f_m^j \circ f_m\right)(x) = f_m^j(2x)$. Now in the $j+1$ case we have $i_{j+1}, x_{j+1}$ with $i_{j+1} < 2^{j-1}$ and $x_{j+1} \in [0, 1)$ such that $2x = 2(i_{j+1} + x_{j+1}) 2^{-j-1} = (i_{j+1} + x_{j+1}) 2^{-j}$. Applying the inductive hypothesis, we obtain that

$$f_m^{j+1}(x) = f_m^j(2x) = \begin{cases} 2x_{j+1} & 0 \leq x_{j+1} \leq 1/2 \\ 2(1-x_{j+1}) & 1/2 \leq x_{j+1} < 1. \end{cases} \tag{31}$$

We conclude the proof by setting $j = h$. Then at each of the $2^h$ values of $x_i = 1/2^h$, we output 0 when $i$ is even since then $x_i = (i/2 + 0) 2^{-h}$ and 1 when $i$ is odd since $x_i = ((i-1)/2 + 1/2) 2^{-h}$. □

*Theorem 5.1.* Note for the upper bound, any $f_m^h \in F_{A_{2,2;h}} \subseteq F_{A_{2h,2}}$. The construction of $A$ follows immediately from Lemma 5.5.

To see that every network $B$ must have error at least $1/6$, we combine Lemmas 5.3 and 5.4. Take $l, w$ such that $w \leq 2^{(h-3)/l-1}$. Then $F_B$ is at most

$$(2w)^l \leq \left(2^{(h-3)/l-1}\right)^l \leq 2^{h-3} \tag{32}$$

-sawtooth by Lemma 5.3. Thus we can bound $t$ in Lemma 5.4 to see that for $f \in F_B$, we have

$$\mathcal{R}(f) \geq \frac{m-4t}{3m} \geq \frac{2^h - 4\left(2^{h-3}\right)}{3\left(2^h\right)} = \frac{2^{h-1}}{6\left(2^{h-1}\right)} = \frac{1}{6}. \tag{33}$$

□

This proof illustrates how composition is able to create many more linear regions than addition. This tool is used in the construction of the deep network as well as the proof that the shallow network cannot create enough oscillations.

A more general theorem is proposed in [2] for the benefits of network depth.

**Theorem 5.6** ([2], Theorem 1.1)**.** *For any positive integer $k$, there exist neural networks of depth $\Theta\left(k^3\right)$ and width $\Theta(1)$ with $\Theta(1)$ distinct parameters that cannot be approximated with $\mathcal{O}(k)$ layers unless they have $\Omega\left(2^k\right)$ width.*

The proof of this theorem is based on the similar idea of network output oscillations. The result is more general that includes piece-wise polynomial functions, convolutional networks and even boosted decision trees. Since many long new definitions are needed (e.g. semi-algebraic gates), we refer people to the proof in the original papers [2].

## 6 The Efficiency of Network Width

Although there is a growing body of work about the efficiency of network depth, Lu et al. in [15] asks an inverse question. Do some wide and shallow networks need exponentially larger deep networks to represent the same functionality?

**Theorem 6.1** ([15] Theorem 1)**.** *This theorem is also known as the universal approximation theorem for width-bounded ReLU networks. For any Lebesgue-integrable function $f : \mathbb{R}^d \mapsto \mathbb{R}$ and any $\epsilon > 0$, there exists a fully-connected ReLU network $A$ with width $k \leq d + 4$, such that the function $F_A$ represented by this network satisfies*

$$\int_{\mathbb{R}^d} |f(x) - F_A(x)| dx < \epsilon. \tag{34}$$

Some concerns about his theorem have been raised by Brandfonbrener [3]. He showed that the $d+4$ is not optimal, and provided a more elegant proof. The following lemma is due to [16] and [5].

**Lemma 6.2.** *Let $\Omega \subset \mathbb{R}^d$ be compact and $f : \Omega \mapsto \mathbb{R}$ be a function computed by a ReLU network with one hidden layer of width $m$. So, we have that $f(x) = \sum_i^n c_i \sigma(a_i^\circ x + b_i)$, where $\sigma$ is the ReLU function, and for each $i$ we have $a_i \in \mathbb{R}^d$ and $b_i, c_i \in \mathbb{R}$. Then, there exists a ReLU network with $n + 1$ hidden layers of width $d + 2$ that computes $f$.*

*Proof.* Since $\Omega$ is bounded, we can find $T > 0$ such that $T + \sum_{i=1}^j c_i \sigma(a_i^\intercal x + b_i) > 0$ for all $j = 1, \cdots, n$, and $T + x_k > 0$ for all $k = 1, \cdots, d$. This value will allow us to avoid mapping inputs to 0 by repeatedly applying the ReLU function.

For each $i$, we can define $d_i \in \mathbb{R}$ such that $a_i^\intercal (x + T\mathbf{1}) + d_i = a_i^\intercal x + b_i$, just by letting $d_i = b_i - a_i^\intercal T\mathbf{1}$. $\mathbf{1}$ is the vector of all value 1. Let $A_i$ be the function calculated by the $i$-th layer. We have that $A_1 : \mathbb{R} \mapsto \mathbb{R}^{d+2}$, $A_i : \mathbb{R}^{d+2} \mapsto \mathbb{R}^{d+2}$ for $i = 1, \cdots, n+1$, and the output layer is $A_{n+2} : \mathbb{R}^{d+2} \mapsto \mathbb{R}$. Take $x \in \mathbb{R}^d$ and $y, z \in \mathbb{R}$, then

$$\begin{aligned}
A_1(x) &= \sigma(x + T\mathbf{1}, a_1^\intercal x + b_1, T) \\
A_2(x, y, z) &= \sigma(x, a_2^\intercal x + d_2, z + c_1 y) \\
&\vdots \\
A_i(x, y, z) &= \sigma(x, a_i^\intercal x + d_i, z + c_{i-1} y) \\
&\vdots \\
A_{n+1}(x, y, z) &= \sigma(x, 0, z + c_n y) \\
A_{n+2}(x, y, z) &= z - T.
\end{aligned} \tag{35}$$

By forming these layers to build the network, we can get the results we want. We are storing the input vector $x$ (with some correction in the case of negative components using $T$) using the first $d$ component of each layer. We then use the $d + 1$-th component to compute the ReLU of the $i$-th affine transformation and $d + 2$-th to compute the $n$ dimensional linear combination of ReLUs, one term at a time. In this way, we reconstruct the entire computational function $f$ for the original wide shallow network. □

With Lemma 6.2, we can now translate any result about the approximation ability of a ReLU network with one hidden layer into a result about a ReLU network with fixed width $d + 2$.

# 7 Conclusion

The goal of this report is to generalize the concept of neural network expressive power. We did a literature review and summarized in our own way the logical flow of how to study the expressive power of neural networks. We investigated why a neural network could be viewed as a universal approximator with a fixed width or depth. We listed several methods for examining the expressive power of neural networks. We systematically studied how network depth was conducive to expressive power. Finally, we also analyzed the effect of network width on expressive power, reinforcing the fact that depth may be more effective than width.

# References


[1] Matus Telgarsky. Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*, 2015.

[2] M Telgarsky. Benefits of depth in neural networks. arxiv 2016. *arXiv preprint arXiv:1602.04485*.

[3] DAVID BRANDFONBRENER. A report on "the expressive power of neural networks: A view from the width". 2017.

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

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

[6] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[7] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8(1):143–195, 1999.

[8] Haim Brezis. *Functional analysis, Sobolev spaces and partial differential equations*. Springer Science & Business Media, 2010.

[9] Elias M Stein and Rami Shakarchi. *Real analysis: measure theory, integration, and Hilbert spaces*. Princeton University Press, 2009.

[10] Leonardo Ferreira Guilhoto. An overview of artificial neural networks for mathematicians.

[11] Raghu Maithra, Poole Ben, Kleinberg Jon, Ganguli Surya, and Sohl-Dickstein Jascha. On the expressive power of deep neural networks. *arXiv preprint arXiv:1606.05336v6*, 2017.

[12] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. *Advances in neural information processing systems*, 27:2924–2932, 2014.

[13] Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *J. Mach. Learn. Res.*, 20:63–1, 2019.

[14] David Brandfonbrener. The expressive power of neural networks cpsc 490. 2018.

[15] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. *Advances in neural information processing systems*, 30:6231–6239, 2017.

[16] Boris Hanin. Universal function approximation by deep neural nets with bounded width and relu activations. *Mathematics*, 7(10):992, 2019.