# IFT 6085 - Guest Lecture
# Expressivity and Universal Approximation Theorems Part 1

This version of the notes has not yet been thoroughly checked. Please report any bugs to the scribes or instructor.

**Scribes**
**Winter 2019:** Moustafa Elarabi, Kun Ni

**Instructor:** Guillaume Rabusseau

## 1 Summary

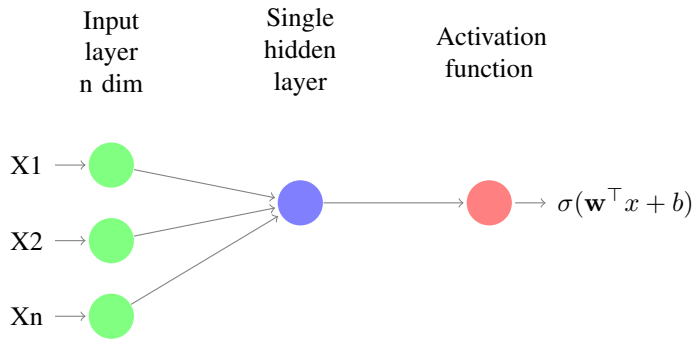
In this lecture we will discuss the expressivity of neural networks, that what kind of functions a neural network could approximate.We will see the limitations of wide neural network and the expressive power of going deeper.

## 2 Introduction

### 2.1 Capacity of Single Perceptron

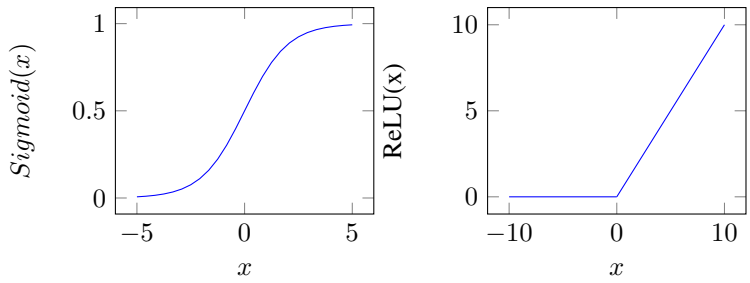
What kind of function can a neural network represent ?

Assuming we have neural model with single hidden layer having 1 hidden unit with activation function $\sigma$, the input layer is $x_i \in \{x_1, x_2, \ldots, x_n\}$, the weight of neuron has the dimension of $1 \times n$



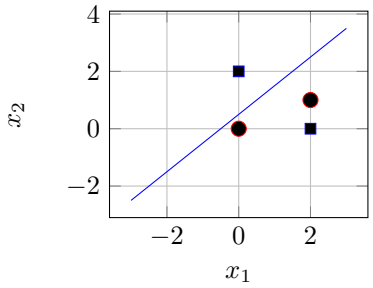

this network have an activation function $\sigma(\sum_i^n \mathbf{w_i} x_i + b) = \sigma(\mathbf{w}^\top x + b)$ and $\sigma$ can be

1. binary $\sigma(t) = Sign(t)$
2. Sigmoid $\sigma(t) = \frac{1}{1+e^{-t}}$
3. ReLU $\sigma(t) = max(0, t)$



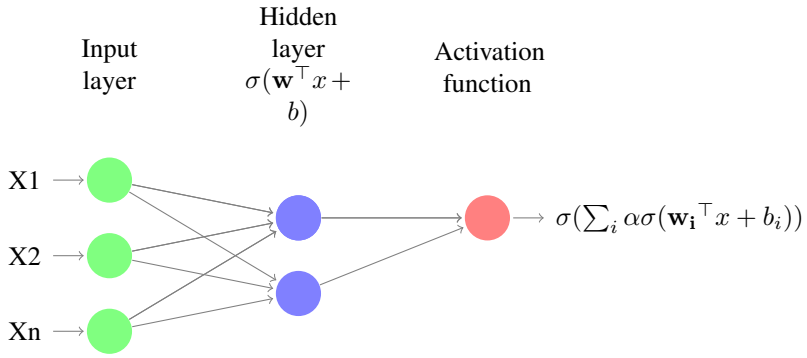

Assuming we are trying to approximate XOR function using the above neural network as discussed in [7].

From the following plot we can see that the XOR function is not linearly separable, hence the model can't interpolate that function well.




### 2.2 Capacity of multiple neurons

A neural model can interpolate the XOR function in case it got a different input representation learnt by another neuron, hence we can approximate this function using multiple hidden units within the same layer.




## 3 Universal Approximation Theorem

The universal approximation theorem looks like any continuous function $f : [0, 1]^n \to [0, 1]$ can be approximated by [some architecture] under [some condition] , which we are going through in the next subsections.

### 3.1 Kolmogorov Arnold Representation theorem

The Kolmogorov-Arnold representation theorem (or superposition theorem) [5] states that every multivariate continuous function can be represented as a superposition of continuous functions of two variables.

It solved a more general form of Hilbert's thirteenth problem [1] which was just questioning if there's an answer for a $7^{th}$ degree equation using continuous functions.
Andrey Kolmogorov and Vladimir Arnold showed that if f is a multivariate continuous function, then f can be written as a finite composition of continuous functions of a single variable and the binary operation of addition.

**Theorem 1.** *Any continuous function* $f : [0,1]^n \to \mathbb{R}$ *can be written as*

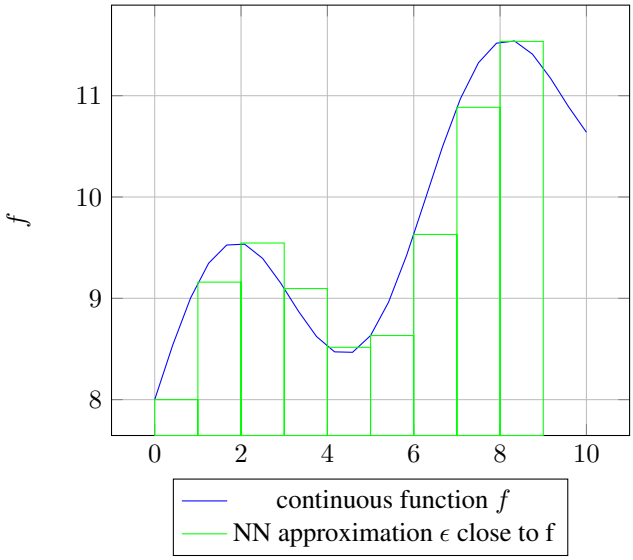
$$f(x) = f(x_1, .., .., x_n) = \sum_{q=1}^{Z_m} \phi_q(\sum_{q=1}^{m} \Psi_q(x_q))$$

### 3.1.1 Comparing with 2 layer NN

The output of a 2 layer Neural network can be represented by the following equation

$$f(x) = \sigma(\sum_i \alpha_i \sigma(\mathbf{w_i}^\top x + b_i))$$

.
Informal theorem: any continuous function $f : [0,1]^n \to [0,1]$ can be approximated to an arbitrary precision by a 2 layer network with Sigmoid activation.

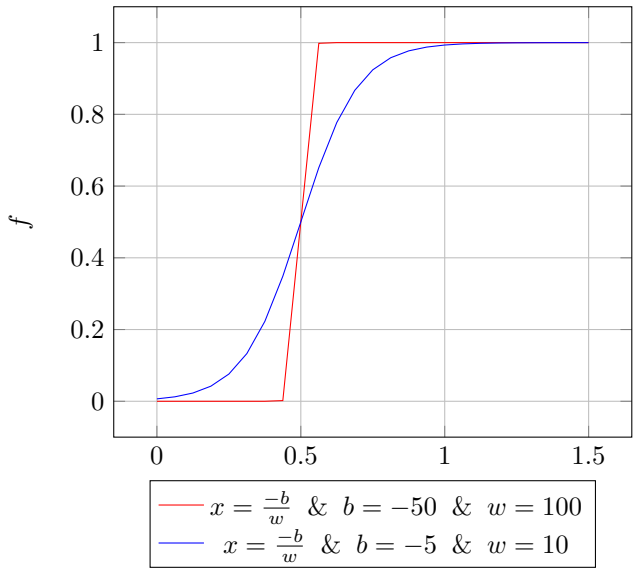



In order to get that neural network approximating the previous continuous function we would have to do the following steps .

- Step 1:

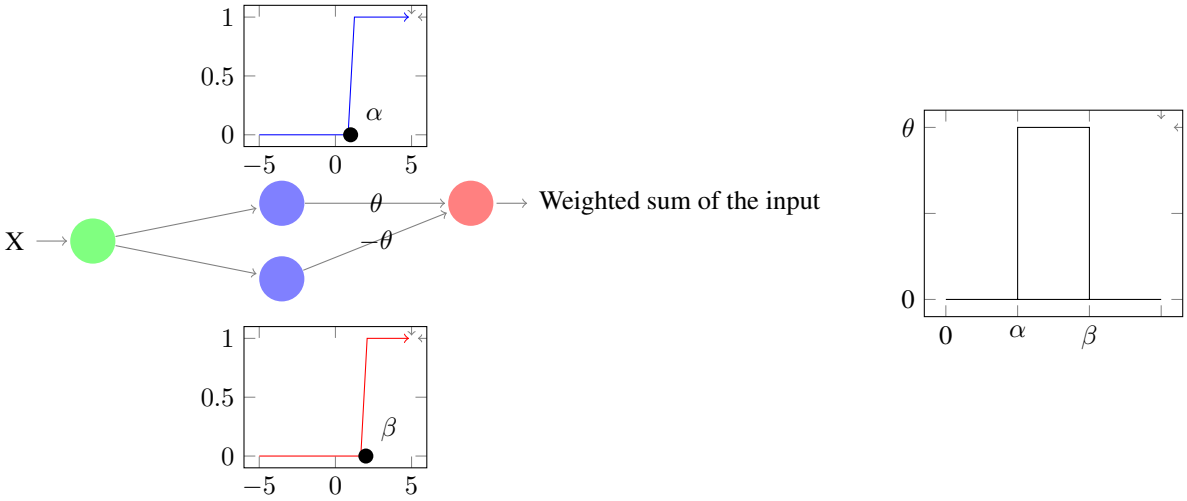
  each single neuron function is $\sigma(\mathbf{w}x + b)$

  playing with $b$ values and $\mathbf{w}$ to get a steep histogram bin used in the approximation



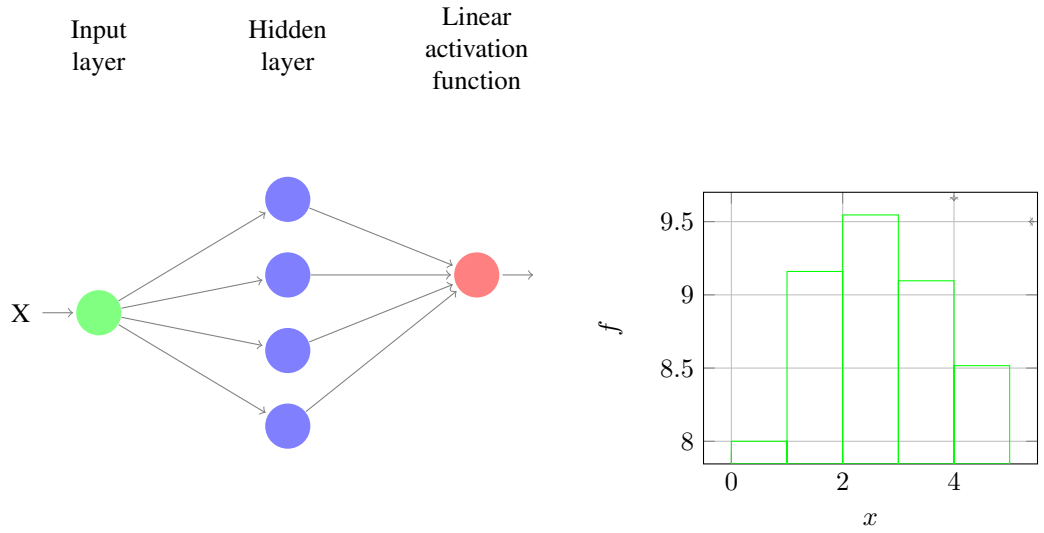

- Step 2 Build 1 block of the histogram

  In this step we are trying to build 1 block of the histogram (used to approximate $f$). The weight of last neuron($\theta$)is use to control the height of the block of histogram. For the first connection, the weight is $\theta$, for the second connection, the weight is $-\theta$. Here we assume $\alpha < \beta$ and the last neuron is linear one. So the last neuron only fires between $\alpha, \beta$. Then we can get the histogram block with height of $\theta$




- Step 3 Discretize $f$

  Neurons of our neural network can represent a single histogram bin trying to approximate the target function with $\epsilon$ error, to reduce that $\epsilon$ we just need to add more neurons




This model is using a linear activation function in the output layer,therefore we will get directly the combined bins of different neurons showing the histogram approximating the target function.

**what if i don't want linear neurons in the output layer ?** The above network's output layer is linear giving us the histogram approximating $f$, adding a sigmoid activation function on the output with same input as above network won't approximate $f$, to get the model to approximate the target function then in this case we can try to approximate $\sigma^{-1} \circ f$ instead of $f$ which using the same above construction will give us the histogram approximating the target function $f$.

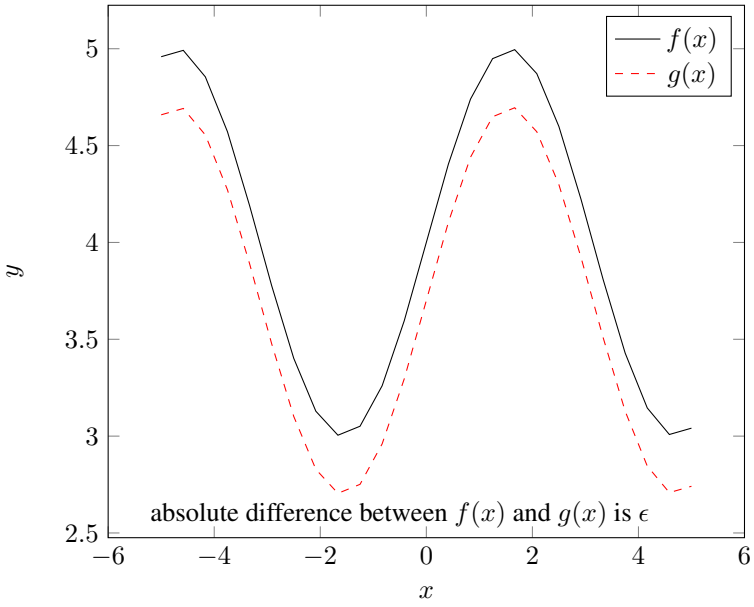
## 3.2 Cybenko Approximation by Superposition of Sigmoidal Function

In this theorem Cybenko [2] proved the universal approximation theorem for artificial neural networks with sigmoid activation functions.

**Theorem 2.** *Let $\sigma$ be any discriminatiry activation function then finite sum of the form $f(x) = \sum_{i=1}^{n} \alpha_i \sigma(\mathbf{w_i}^\top + b_i)$ is dense in $C([0,1]^n)$*

$C([0,1]^n)$ is a set of continuous function $[0,1]^n \to \mathbb{R}$ and discriminatory or sigmoidal activation functions means $\sigma(t) = 0$ as $t \to -\infty$ and $\sigma(t) = 1$ as $t \to \infty$.

**Informal explanation of the theorem** this theorem means that for any $g \subset C([0,1]^n)$ and any $\epsilon > 0$, there exists $f : x \to \sum_{i=1}^{n} \alpha_i \sigma(\mathbf{w_i}^\top + b)$ that $|f(x) - g(x)| < \epsilon$ for all $x \subset [0,1]^n$.




**Proof** a similar result was independently obtained by Hornik[4] and also by Funahashi[3] using different tools. Hornik's proof relies on the Stone-Weierstrass Theorem which states that every continuous function defined on a closed interval $[a, b]$ can be uniformly approximated as closely as desired by a polynomial function.

### 3.3 Expressive Power of Deep Neural Network lu et el

The expressive power of deep neural networks a view from the width by Lu et el 2017 [6]

**Theorem 3.** *Let $n$ be the input dimension. For any integer $k >= n + 4$ there exists $F_\alpha : \mathbf{R^n} \to \mathbf{R}$ represented by a relu neural network $\alpha$ with width $d_m = 2k^2$ and depth $h = 3$ such that for any constant $b > 0$,there exists an $\epsilon > 0$ and for any function $F_\beta : \mathbf{R^n} \to \mathbf{R}$ represented by a relu neural network $\beta$ whose parameters are bounded in $[-b, b]$ with width $d_m <= k^{\frac{3}{2}}$ and depth $h <= k + 2$ the following inequality holds*

$$\int_{\mathbf{R}} |F_\alpha - F_\beta| dx >= \epsilon$$

This theorem states that there are networks such that reducing width requires increasing in the size to compensate, which is similar to that of depth qualitatively.

#### 3.3.1 Result

1. Any continuous function $f : \mathbf{R^n} \to \mathbf{R}$ can be approximated by ReLU network with width bounded by $n + 4$ where n is the input dimension.
2. Moreover, except for a measure zero set, all functions cannot be approximated by width-n ReLU networks

## 4 Expressive of Deep neural network

### 4.1 Telgarsky 2015 theorem

In this section, We want to show interesting results from [8]. Why we choose depth over width. $f$ can be computed by a deep NN with m neurons but any shallow NN computing $f$ has at least $\Omega(2^m)$ neurons.That is, shallow network

needs exponential number of neurons to get the similar result as deep network.

Here, we introduce a specific family of classification problem, indexed by a positive integer $k$, all shallow networks with fewer than exponentially(in $k$) nodes exhibit at least 1/6 classification error. But to a deep network with 2 nodes in each of $2k$ layers achieves zero error. The proof is elementary, and the networks are standard feedforward networks with ReLU activation.

**Theorem 4** (The representation power of the deep NN)**.** *For any positive integer $k$, $\exists$ a set of $n = 2^k$ samples in $S = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ where $x_i \in [0, 1], y \in \{0, 1\}$ s.t.*

1. *$\exists$ a ReLU NN with 2k layers, 2 neurons in each layers archiving zero classification errors in the sample set S*
2. *$\forall$ ReLU NN with $l$ layers and $m \leq 2^{\frac{k-3}{l-1}}$ neurons in each layers misclassify at least 1/6 of points in S*

### 4.1.1 example

Here is the example of comparing deep and shallow networks.

**Example 5** (Deep and shallow networks)**.**

*For $k = 50$,*

*$\exists$ a deep ReLU NN (100 layers, 200 neurons) having zero error.*

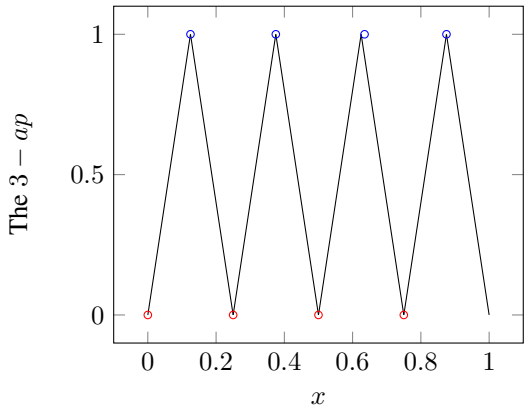
*While any 2 layers ReLU NN with less than $2^{47}$ neurons misclassify at least 1/6 points in S.*

*That is the benefits of depth*

## 4.2 Proof

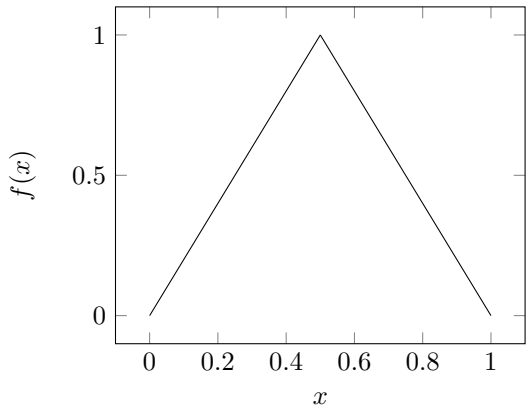
### 4.2.1 Classification Problem

Let n-ap (the n-alternating-point problem) denote the set of n uniformly spaced points within $[0, 1 - 2^{-n}]$ with alternating labels, as depicted in Figure 1; that is, the points $((x_i, y_i))_{i=1}^n$ with $x_i = i2^n$, and $y_i = 0$ when i is even, and otherwise $y_i = 1$. As the $x$ values pass from left to right, the labels change as often as possible.

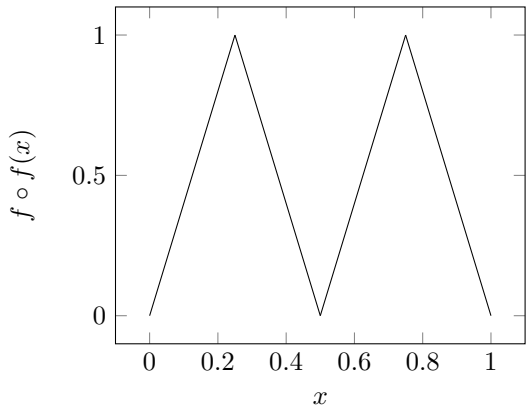


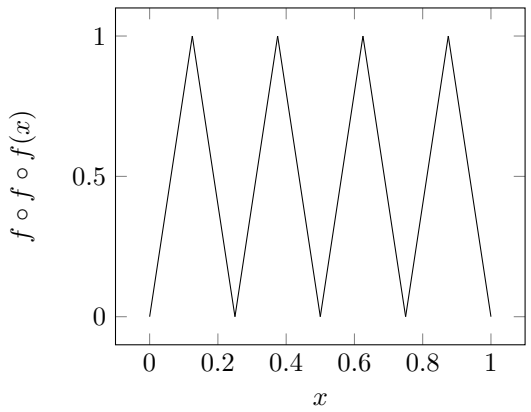
### 4.2.2 Sawtooth function

We can define $f$ as figure shows:




$f$ can be computed by a ReLU NN with 2 layers and 3 neurons ReLU network. For instance, this network: $f(x) = \sigma_R(2\sigma_R(x) - 4\sigma_R(x - 1/2))$. It can be applied in $k$ times, for example: $f \circ f$, and $f \circ f \circ f$, Here are the figures. Their peaks and troughs match the $2^2$-ap and $2^3$-ap




**Definition 6** (Sawtooth)**.** *A function $f : \mathbb{R} \to \mathbb{R}$ is $t$-saw-tooth if it is piece-wise linear with $t$-pieces*

This means $\mathbb{R}$ is partitioned into $t$ consecutive intervals.

**Lemma 1.** *If $f$ is $t$-sawtooth, $g$ is $s$-sawtooth, then wen can get*

$$f + g \; is \; (s+t) - sawtooth$$

$$f \circ g \; is \; st - sawtooth$$

**Lemma 2.** *If $\sigma$ is $t$-sawtooth, then any NN with $\sigma$ activation, and $l$ layers and $m$ neurons is $(tm)^l - sawtooth$*

**Lemma 3.** *If $f$ is $t$-sawtooth, then f misclassify at least $\frac{n-4t}{3n}$ points in S, here $n = 2^k$*

so the proof of theorem is straightforward now. Because $\sigma_R$ is 2-sawtooth. and $m \leq 2^{\frac{k-3}{l-1}}$, then the lower bound on classification error is

$$\frac{n - 4(tm)^l}{3n} = \frac{1}{3} - \frac{3}{4}(2m)^l 2^{-k} \geq \frac{1}{3} - \frac{3}{4}(2)^{k-3}2^{-k} = \frac{1}{3} - \frac{1}{6} = \frac{1}{6}$$

# References


[1] S. S. ABHYANKAR. Hilberts thirteenth problem. In *Proc. of the Franco-Belgian Conference in Reims, Société Mathématique de France, Séminaires et Congres*, volume 2, page 1, 1997.

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

[3] K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2 (3):183–192, 1989.

[4] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[5] A. N. Kolmogorov. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. In *Doklady Akademii Nauk*, volume 114, pages 953–956. Russian Academy of Sciences, 1957.

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

[7] M. Minsky and S. Papert. Perceptrons: an introduction to computational geometry. 1969.

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