On the Size and Width of the Decoder of a Boolean Threshold Autoencoder

Tatsuya Akutsu®, Senior Member, IEEE, and Avraham A. Melkman®

Abstract—In this brief paper, we study the size and width of autoencoders consisting of Boolean threshold functions, where an autoencoder is a layered neural network whose structure can be viewed as consisting of an encoder, which compresses an input vector to a lower dimensional vector, and a decoder which transforms the low-dimensional vector back to the original input vector exactly (or approximately). We focus on the decoder part and show that \( O(\sqrt{Dn}) \) and \( O(\sqrt{Dn}) \) nodes are required to transform \( n \) vectors in \( d \)-dimensional binary space to \( D \)-dimensional binary space. We also show that the width can be reduced if we allow small errors, where the error is defined as the average of the Hamming distance between each vector input to the encoder part and the resulting vector output by the decoder.

Index Terms—Autoencoders, Boolean functions, neural networks, threshold functions.

I. INTRODUCTION

Extensive studies have been done on artificial neural networks not only from a practical viewpoint but also from a theoretical viewpoint [1], [2], [3]. Among various models of neural networks, autoencoders have recently attracted much attention due to their ability to generate new data and have been applied to various areas including image processing [4], [5], natural language processing [5], and drug discovery [6]. A comprehensive survey on variants and applications of autoencoders can be found in [7]. An autoencoder is a layered neural network consisting of two parts, an encoder and a decoder, where the former transforms an input vector to a low-dimensional vector and the latter transforms the low-dimensional vector to an output vector which should be the same as or similar to the input vector [8], [9], [10], [11]. Therefore, an autoencoder maps input data to a low-dimensional representation space. Such a mapping is obtained via unsupervised learning that minimizes the difference between input and output data by adjusting weights (and some other parameters).

As mentioned above, autoencoders perform dimensionality reduction, a kind of data compression. Indeed, autoencoders have been applied to the compression of image data [12], [13], [14]. In order to understand the dimensionality reduction and data compression mechanisms of autoencoders, several studies have been done. Baldi and Hornik [9] studied relations between principal component analysis and autoencoders with one hidden layer. Hinton and Salakhutdinov [10] empirically studied relations between the depth of autoencoders and the dimensionality reduction, the results of which suggest that deeper networks can produce lower reconstruction errors.

Kärkkäinen and Hänninen [15] also conducted empirical studies on relations between the depth and the dimensionality reduction using a variant model of autoencoders, and concluded that deeper networks obtain lower autoencoding errors during the identification of the intrinsic dimension, but the detected dimension does not change compared to a shallow network. Recently, several studies have been done to understand information flow in autoencoders by analyzing mutual information between layers [16], [17], [18].

However, how data are compressed via autoencoders is still unclear. In particular, to the best of the authors’ knowledge, there had been almost no theoretical result on the quantitative relationship between the compressive power and the size and depth in autoencoders although extensive studies have been done on the quantitative relationship between the representational power and the size and depth in various types of layered neural networks [19], [20], [21], [22].

In order to study the relationship in autoencoders, Melkman et al. [23] analyzed the relations between the architecture (e.g., the numbers of nodes and layers) of networks and their compression ratios, using a layered Boolean threshold network (BTN), which is a discrete model of neural networks. A BTN is equivalent to a threshold circuit [1], [2], [3] in which each node takes on values that are either 1 (active) or 0 (inactive) and the activation rule for each node is given by a Boolean threshold function. In [23], several architectures of autoencoders were presented that map \( n \) \( D \)-dimensional binary input vectors into \( d \)-dimensional binary space and then recover the original input vectors. In particular, they showed the following architectures for \( d = \lceil \log n \rceil \) (or, almost equivalently, \( d = 2\lceil \log n \rceil \)): a four-layer encoder with \( O(\sqrt{n} + D) \) nodes; a five-layer autoencoder with \( D/n/D/n/D \) architecture, where each parameter means the number of nodes in the corresponding layer; a seven-layer autoencoder with \( O(D\sqrt{n}) \) nodes; and a decoder with depth \( n + 1 \) and width \( O(D) \), where the width is the maximum number of nodes per layer (except for the input and output layers) and the depth is the number of layers minus one. However, it was unclear whether or not these results are optimal (or near optimal).

In this brief paper, we focus on the decoder part because the decoders (in the above-quoted results) use \( O(\sqrt{n} + D) \), \( O(n + D) \), or more nodes whereas the encoder uses only \( O(\sqrt{n} + D) \) nodes. We show that \( O(\sqrt{n}/d^{1/2}) \) lower bound on the size of the perfect decoder, where a decoder is called perfect if it can always recover the original input vectors exactly. To the authors’ knowledge, this is the first lower bound result on the size (i.e., the number of nodes) of the autoencoder.

As a positive result, we show that there exists a perfect decoder with width \( \max((n/B)^{1/2} + B, BD) \) and a constant depth, where \( B \) is an arbitrary integer larger than 1. By letting \( B = \lceil (n/D)^{1/2} \rceil \), we obtain an \( O(\sqrt{Dn}) \) upper bound on the size and width of the decoder, which is relatively close to...
the lower bound and improves the previous $O(D\sqrt{n})$ upper bound. In order to construct decoders that have even smaller width, we permit them to output vectors that are in error, where the error is defined as the average of the Hamming distance between each vector input to the encoder part and the resulting vector output by the decoder. We show that there exists a decoder with width $\max([n/B]+B,(B-1)D+1)$ and a constant depth whose error is at most $D((1/B2^B)+(1/n))$.

II. PRELIMINARIES

A function $f: \{0, 1\}^i \rightarrow \{0, 1\}$ is called a Boolean threshold function if it is represented as

$$f(x) = \begin{cases} 1, & \mathbf{a} \cdot \mathbf{x} \geq \theta \\ 0, & \text{otherwise} \end{cases}$$

for some ($\mathbf{a}, \theta$), where $\mathbf{a}$ is an $h$-dimensional integer vector and $\theta$ is an integer. We will also denote the same function as $[\mathbf{a} \cdot \mathbf{x} \geq \theta]$. An acyclic network is called a BTN if all activation functions in the network are Boolean threshold functions.

In this brief paper, we only consider layered BTNs in which nodes are divided into $L$-layers and each node in the $i$th layer has inputs only from nodes in the $(i-1)$th layer ($i = 1, \ldots, L-1$). Then, the states of nodes in the $i$th layer can be represented as a $W_i$-dimensional binary vector, where $W_i$ is the number of nodes in the $i$th layer and is called the width of the layer. A layered BTN is represented as $y = f^{L-1}(f^{L-2}(\ldots f^1(x)\ldots))$, where $x$ and $y$ are the input and output vectors, respectively, and $f^0$ is a list of activation functions for the $(i+1)$th layer. The 0th and $(L-1)$th layers are called the input and output layers, respectively, and the corresponding nodes are called input and output nodes, respectively. The size, depth, and width of a BTN are defined as the number of nodes ($\sum_{i=0}^{L-1} W_i$), the number of layers minus one ($L - 1$), and the maximum number of nodes in a layer other than the input and output layers ($\max(W_i: 0 < i < L - 1)$), respectively. When we consider autoencoders, one layer ($4$th layer where $k \in \{1, \ldots, L - 2\}$) is specified as the middle layer, and the nodes in this layer are called the middle nodes (see also Fig. 1). Then, the middle vector $z$, encoder $f$, and decoder $g$ are defined by

$$\begin{align*}
z &= f^{L-1}(f^{L-2}(\ldots f^1(x)\ldots)) = f(x) \\
y &= f^{L-1}(f^{L-2}(\ldots f^1(z)\ldots)) = g(z).
\end{align*}$$

Since the input vectors should be recovered from the middle vectors at the output layer, we assume that $x$ and $y$ are $D$-dimensional binary vectors and $z$ is a $d$-dimensional binary vector with $d \leq D$. A list of functions is also referred to as a mapping. The $i$th element of a vector $x$ will be denoted by $x_i$, which is also used to denote the node corresponding to this element. Similarly, for each mapping $f$, $f_i$ denotes the $i$th function.

Let $X_n = \{x^0, \ldots, x^{n-1}\}$ be a set of $n$ $D$-dimensional binary input vectors that are all different. We define perfect encoder, decoder, and autoencoder as follows [23].

Definition 1: A mapping $f: \{0, 1\}^D \rightarrow \{0, 1\}^d$ is called a perfect encoder for $X_n$ if $f(x^i) \neq f(x^j)$ holds for all $i \neq j$.

Definition 2: A pair of mappings $(f, g)$ with $f: \{0, 1\}^D \rightarrow \{0, 1\}^d$ and $g: \{0, 1\}^d \rightarrow \{0, 1\}^D$ is called a perfect autoencoder if $g(f(x^i)) = x^i$ holds for all $x^i \in X_n$. Furthermore, such $g$ is called a perfect decoder.

Note that a perfect decoder exists only if there exists a perfect autoencoder. Furthermore, it is easily seen from the definitions that if $(f, g)$ is a perfect autoencoder, $f$ is a perfect encoder.

Example 3: Let $X_4 = \{x^0, x^1, x^2, x^3\}$, where $x^0 = (0, 0, 0)$, $x^1 = (1, 0, 0)$, $x^2 = (1, 0, 1)$, and $x^3 = (1, 1, 1)$. Let $D = 3$ and $d = 2$. Define $z = f(x)$ and $y = g(z)$ by

$$\begin{align*}
f_0(x) &= [x_0 + x_1 - x_2 \geq 1], & f_1(x) &= [x_2 \geq 1] \\
g_0(z) &= [z_0 + z_1 \geq 1], & g_1(z) &= [z_0 + z_1 \geq 2] \\
g_2(z) &= [z_1 \geq 1].
\end{align*}$$

This pair of mappings has the following truth table, which shows it to be a perfect autoencoder.

| $x_0$ | $x_1$ | $x_2$ | $z_0$ | $z_1$ | $y_0$ | $y_1$ | $y_2$ |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |
| 1     | 0     | 0     | 0     | 1     | 0     | 1     | 0     |
| 1     | 0     | 1     | 0     | 1     | 1     | 1     | 0     |
| 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     |

III. LOWER BOUNDS ON THE SIZE OF THE DECODER

In this section, we derive lower bounds on the number of threshold units (i.e., the number of nodes except those in the middle layer) in the decoder BTN of a Boolean threshold autoencoder which perfectly encodes every possible set of $n$ $D$-dimensional vectors, and which has a middle layer whose size is less than one-third of the size of the output layer. Note that $n \leq 2^d$, since $n$ vectors are perfectly encoded.

Denote the number of threshold units of the decoder BTN by $N$, with the $i$th unit having $d_i$ inputs, where $N$ includes the number of output nodes but does not include the number of middle nodes.

Theorem 4: Suppose that for given parameters $n$, $D$, and $d \leq ((D-1)/3)$, there exists a perfect autoencoder. Then, the number of threshold units of its (necessarily perfect) decoder BTN, $N$, satisfies $N \geq \sqrt{n}/(d + (1/d))$.

Proof: The idea is to calculate an upper bound on the total number of different sets of $n$ vectors that can be generated at the output layer of the autoencoder from all possible sets of $d$-dimensional vectors, input to the middle layer, by all possible different decoder BTNs. Let us denote that number $PD$ (the in-principle decoded sets). Since the autoencoder is perfect, $PD$ is not smaller than the number of different sets of $D$-dimensional output (=input) vectors of the autoencoder, each of size $n$, which is $(2^D/n)^n \geq (2^D-n)^n/n^n$.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Next we derive an upper bound on $PD$. Each threshold unit computes a Boolean function of the $d$ inputs (in the middle layer). In the course of proving [1, Theorem 7.4], which provides a lower bound on the size of the universal networks, it is shown that the number of different functions representable by a threshold unit with in-degree $d_i$ in an acyclic BTN which has $d$ input nodes is at most $2^{dd+1}$. Since the decoder is acyclic, we assume, without loss of generality, that its threshold units are topologically sorted so that $d_i ≤ d+i, 0 ≤ i ≤ N−1$. Therefore, $\sum_{i=0}^{N−1} d_i ≤ dN + (1/2)N^2$, and an upper bound on the total number of functions from $\{0, 1\}^d$ to $\{0, 1\}^D$ computable by a decoder BTN with $N$ nodes is

$$\prod_{i=0}^{N−1} 2^{dd+1} = 2^{N+d\sum_{i=0}^{N−1} d_i} < 2^{N+d^2N+\frac{3}{2}N^2}.$$ (1)

Observe that the number of all possible ordered sets of $n$-dimensional vectors in the middle layer does not exceed $2^{m}\cdot|D^n|$. We conclude from the foregoing that

$$\frac{(2D−n)^n}{n^n} ≤ PD ≤ \left(\frac{2D}{n}\right)^n \left(2^{N+d^2N+\frac{3}{2}N^2}\right).$$

Taking base 2 logarithms of both sides shows that the parameters of the decoder satisfy $nd + N + d^2N + (d/2)N^2 ≥ n \log(2D−n)−n \log n$. Noting that $n ≤ 2^d ≤ 2^{D−1}$ yields $N + d^2N + (d/2)N^2 > n(D−1)−2nd$. Applying the pre-condition $d ≤ ((D−1)/3)$ to this inequality results in

$$N + d^2N + \frac{d}{2}N^2 > \frac{n}{3}(D−1).$$

By using the quadratic solution formula, we find that

$$N > \frac{−(1 + d^2) + \sqrt{(1 + d^2)^2 + 2dM(D−1)}}{d} \geq \frac{2}{\sqrt{3d}}(D−1)\sqrt{n} − \left(d + \frac{1}{d}\right),$$

from which the theorem follows.

**Remarks:**

1. This lower bound is meaningful only if $((2/3d)(D−1))^{1/2}\sqrt{n} − (d + (1/d)) ≥ D$ because the decoder contains $D$ output nodes.
2. The lower bound stated in the Theorem holds for any acyclic decoder (not just for layered ones) because $d_i ≤ d+i−1$ for any such decoder.
3. The stated lower bound implies an $\Omega((Dn/d)^{1/2})$ lower bound on the widths of layered BTN decoders of constant depth.

As mentioned in the above, Theorem 4 holds for any acyclic decoder. On the other hand, we focus on layered decoders in this brief paper. By making use of the layered architecture, the lower bound can be improved by a constant factor [excluding a negligible factor of $(1/d)$] by tightening the inequality (1).

As a first step, we derive a tighter bound on the value of $\sum_{i=0}^{N−1} d_i$, the number of edges in the decoder. To that end, we introduce some notations. Let $\mathcal{L}_M(d, D)$ be the set of layered decoders with $d$ units in the input layer, $D$ units in the output layer, and $M$ internal units. For $G \in \mathcal{L}_M(d, D)$ denote by $E(G)$ the set of edges of $G$, and let

$$e = \max\{|E(G)|: G \in \mathcal{L}_M(d, D)\}.$$

**Lemma 5:** If $M ≥ 2(d + D)$, then any $G$ that maximizes $|E(G)|$ has two inner layers, and

$$e = \begin{cases} \frac{1}{4}M^2 + \frac{1}{2}M(d + D) + \frac{1}{4}\delta^2, & \text{if } M−\delta \text{ is even} \\ \frac{1}{4}M^2 + \frac{1}{2}M(d + D) + \frac{1}{4}\delta^2 − 1, & \text{if } M−\delta \text{ is odd.} \end{cases}$$

In particular, $e ≤ (1/4)M^2 + (1/2)M(d + D) + (1/4)\delta^2$.

**Proof:** Suppose $G \in \mathcal{L}_M(d, D)$ has $L > 2$ inner layers, with $M_i$ units in its $i$th layer, where $M_0 = d, M_{L+1} = D$. We will show that there is a $G' \in \mathcal{L}_M(d, D)$ with $L − 1$ inner layers such that $|E(G)| ≤ |E(G')|$. Set $M_i' = M_i + 3$ for $3 ≤ i ≤ L$, $M_0' = d$ and distinguish two cases for the number of units in layers 1 and 2.

1. If $M_2 ≥ M_3$, set $M_1' = M_1 + M_3, M_2' = M_2$.
   Then $|E(G')| − |E(G)| = dM_1 + M_2M_4 − M_3M_4 > 0$.
2. If $M_2 < M_3$, set $M_1' = M_1 + M_2, M_2' = M_2$.
   Then $|E(G')| − |E(G)| = dM_1 + M_3M_4 − M_1M_2 > 0$.

Therefore, when looking for a $G$ having the largest number of edges among $G \in \mathcal{L}_M(d, D)$ with at least two inner layers, we can restrict ourselves to $G$ with two inner layers. We show next that the first inner layer of such an extremal $G$ has either $\lfloor((M−\delta)/2)\rfloor$ or $\lceil((M−\delta)/2)\rceil$ units, where $δ = D−d$.

Suppose $G$ has two inner layers, but $M_1 ≤ \lfloor((M−\delta)/2)\rfloor − 1$. Consider $G'$ such that $M_1' = M_1 + 1$. Then

$$|E(G')| − |E(G)| = M_1 + 1 − 2M_1 ≥ M_1 + 1 − 2\left[\frac{M−\delta}{2}\right] > 0.$$  

Similarly, if $M_1 ≥ \lceil((M−\delta)/2)\rceil + 1$, then $G'$ with $M_1' = M_1 − 1$ and $M_2' = M − M_1 + 1$ satisfies $|E(G')| − |E(G)| > 0$.

Next, we compute the number of edges of a decoder, $G^*$, with the largest number of edges among layered decoders with at least two inner layers. As we just saw, if $M−\delta$ is even (odd), then its first inner layer has $M_1' = (1/2)(M−\delta)$ [resp. $M_1' = (1/2)(M−\delta−1)$] units. Substituting these into $e^* = dM_1 + M_2M_3 + (M−M_1′)D$ shows that

$$e^* = \begin{cases} \frac{1}{4}M^2 + \frac{1}{2}M(d + D) + \frac{1}{4}\delta^2, & \text{if } M−\delta \text{ is even} \\ \frac{1}{4}M^2 + \frac{1}{2}M(d + D) + \frac{1}{4}\delta^2 − 1, & \text{if } M−\delta \text{ is odd.} \end{cases}$$

To complete the proof, we note that if $M ≥ 2(d + D)$, then $e^*$ is greater than the number of edges of the decoder with a single inner layer of size $M$, which is $M(d + D)$.

**Theorem 6:** Suppose that for given parameters $n, D$, and $d ≤ ((D−1)/3)$, there exists a perfect autoencoder with layered architecture having at least one inner layer. Then, the number of threshold units of its (necessarily perfect) decoder-BTN, $N$, satisfies $N > ((4(D−1))/3d)^{1/2}\sqrt{n} − (d + (2/d))$.

**Proof:** We modify the proof of Theorem 4 by refining the upper bound of inequality (1), using Lemma 5

$$N + d \sum_{i=0}^{N−1} d_i ≤ N + de ≤ N + \frac{1}{4}dM^2 + \frac{1}{2}dM(d + D) + \frac{1}{4}d\delta^2.$$
Rephrasing this upper bound in terms of \( N \) as used in Theorem 4, namely, \( N = M + D \), we get
\[
N + \frac{1}{4} d M^2 + \frac{1}{2} d M (d + D) + \frac{1}{4} d^2 s^2
= N + \frac{1}{4} d (N - D)^2 + \frac{1}{2} d (N - D) (d + D) + \frac{1}{4} d (D - d)^2
= N + \frac{1}{4} d N^2 + \frac{1}{2} d^2 N - \frac{1}{4} d^2 (4D - d)
< N + \frac{1}{4} d N^2 + \frac{1}{2} d^2 N.
\]
Therefore, \((n/3)(D - 1) < \log PD < N + (1/2)d^2 N + (1/4)dN^2\).

Using the quadratic solution formula, we get
\[
N > \frac{-\left(\frac{d^2}{2} + 1\right) + \sqrt{\left(\frac{d^2}{2} + 1\right) + \frac{4}{\sqrt{3}} d (D - 1)}}{2}.
\]

**Remark:** It is assumed in the above that the decoder has at least one inner layer. If there is no inner layer, inequality (1) becomes \( \prod_{i=0}^{N-1} 2^{d+1} < 2^{d + d^2} \), and thus, \( D + d^2 D > (n/3)(D - 1) \) must hold. However, this inequality is not satisfied if \( n \gg d^2 \). Therefore, at least one inner layer is required for the perfect decoder with reasonably large \( n \).

**IV. Decoder With Width max([n/B] + B, BD)**

In this section, we present the architecture of a decoder BTN that can decode perfectly any set of \( n \) \( D \)-dimensional vectors that has been encoded into a layer of size \( d \) (so that the encoder and decoder together perfectly autoencode every such set), where \( d = \lceil \log n \rceil \).

We will use \( \beta_0, \ldots, \beta_{d-1} \) to refer to the middle nodes as well as to their values, with \( \beta \) denoting the vector \( (\beta_0, \ldots, \beta_{d-1}) \). \( \text{Int}(\beta) \) denotes the integer encoded by \( \beta \), i.e., \( \text{Int}(\beta) = \sum_{i=0}^{d-1} \beta_i 2^{d-1} \).

Let \( B \) be an integer greater than 1. For ease of exposition, we assume that \( n \) can be divided by \( B \) and use \( n_B \) to denote \( n/B \). Otherwise, we can let \( n_B = \lceil n/B \rceil \). The decoder has \( d/(n/B) + B/BD/D \) architecture, which means that the middle layer, the first and second inner layers, and the output layer have \( d, (n/B) + B, BD, \) and \( D \) nodes, respectively. Precisely, in addition to \( \beta \) nodes in the middle layer, it has the following nodes in the first and second inner layers, and the output layer, respectively (see Fig. 2):

1. \( \gamma_0, \gamma_1, \ldots, \gamma_{n_B - 1}, \gamma_{n_B + 1}, \ldots, \gamma_{n_B + B - 1} \);
2. \( y_{j,b} \) for \( j = 0, \ldots, D - 1 \) and \( b = 0, \ldots, B - 1 \);
3. \( y_j \) for \( j = 0, \ldots, D - 1 \), the output nodes.

The activation function of the \( \gamma_i \) are
\[
\gamma_i = \lceil \frac{\text{Int}(\beta)}{B} \rceil = i, \quad \text{for} \ i = 0, \ldots, n_B - 1
\]
\[
\gamma_{n_B + b} = \lceil \text{Int}(\beta) = h \; \text{(mod} B) \rceil, \quad \text{for} \ h = 0, \ldots, B - 1
\]
where \( \lceil \text{Int}(\beta) = k \rceil \) takes value 1 if \( \text{Int}(\beta) = k \), and value 0 otherwise. Note that division and modulo operations can be done by threshold circuits of width \( O(\text{poly}(\log n)) \) and depth 5 \cite{24} [recall that \( \beta \) is a vector of dimension \( O(\log n) \)], so that the width of our network remains \( O(\sqrt{D} n) \). Furthermore, if \( B = 2^K \) for some integer \( K \), we do not need such circuits because we can simply use the first \( d - K \) bits of \( \beta \) to represent \( \text{Int}(\beta)/B \) and the remaining \( K \) bits to represent \( \text{Int}(\beta) \mod B \). Also note that \( \lceil \text{Int}(\beta) = k \rceil \) can be calculated by using a single unit with an activation function \( \text{a} \cdot \beta \geq \theta \) such that
\[
a_i = \begin{cases} 1, & \text{if } k_i = 1 \\ -1, & \text{otherwise} \end{cases}
\]
\[
\theta = \sum_i k_i
\]
where \( k_0, k_1, \ldots, k_{d-1} \) is the binary representation of an integer \( k \).

The activation function of \( y_{j,b} \) is
\[
y_{j,b} = \sum_{i=0}^{n_B + B - 1} w_{i,j,b} y_i \geq 2
\]
where \( w_{i,j,b} \) is given by
\[
w_{h,j,b} = x_j^{Bh+b}, \quad \text{for } h = 0, \ldots, n_B - 1
\]
for \( b = 0, \ldots, B - 1 \).

Numbering the \( n \) input-output vectors so that each \( x^k \) is mapped to \( \beta \) with \( \text{Int}(\beta) = k \), it is not difficult to verify that when \( x^k \) is input to the autoencoder, with \( k = B\ell + r \), \( 0 \leq r \leq B - 1 \), then \( y_{j,b} = x_j^{B\ell+b} + [b = r] \geq 2 \). Hence, \( y_{j,b} = 1 \) if and only if \( b = r \) and \( x_j^{k} = 1 \).

The activation function of \( y_j \) is
\[
y_j = \sum_{b=0}^{B-1} y_{j,b} \geq 1
\]
which ensures that on input \( x^k \), the value of \( y_j \) is \( x_j^{k} \), i.e., the decoder is perfect.

Although we assumed that \( x^k \) is mapped to \( \beta \) with \( \text{Int}(\beta) = k \), this assumption does not pose any restriction on an encoder.
because arbitrary permutations of input vectors can be considered. Therefore, we have the following.

**Theorem 7:** For any perfect encoder that maps $X_n$ one-to-one to $d$-dimensional binary vectors, with $d \geq \lceil \log n \rceil$ and $n$ sufficiently large, there exists a constant perfect decoder of a constant depth and width at most $\max (\lceil n/B \rceil + B, B D)$, where $B$ is any integer such that $1 < B \leq n$.

Note that if $B = 2^k$ for some integer $K$, we do not need a condition of “sufficiently large $n$” (precisely, $n \gg (\log n)^k$ for some constant $k$). This condition is needed only for calculating $\lceil \text{Int}(\beta)/B \rceil$ and $(\text{Int}(\beta) \mod B)$, the details of which are not relevant to this brief paper.

Upon choosing $B = \lceil (n/D)^{1/2} \rceil$ in Theorem 7, we obtain the following corollary.

**Corollary 8:** Suppose $n > D$. Then, for any perfect encoder that maps $X_n$ one-to-one to $d$-dimensional binary vectors with $d = \lceil \log n \rceil$, there exists a constant depth perfect decoder width $O(\sqrt{Dn})$ nodes.

Interestingly, this corollary improves the $O(D\sqrt{n})$ size of a perfect decoder given in [23], while using a simpler architecture. Furthermore, the order of this upper bound is close to that of the lower bound given in Theorem 4.

V. APPROXIMATE DECODERS

In Section IV, we showed a construction of a perfect decoder of width $\max (\lceil n/B \rceil + B, B D)$. It seems difficult to construct constant depth decoders with smaller width. In this section, we show that it is possible to improve the above bound on the width of such decoders, while retaining a constant depth, if small errors between the original input vectors (to the encoder) and the output vectors are permissible. This relaxation is a reasonable one because the input and output vectors are not necessarily the same in practice. The improvement results from a reduction in the number of $y_{j,b}$ nodes from $BD$ to $(B - 1)D$ by allowing small errors. In Section IV, we used node $y_{j,b}$ to recover $x_i^k$ for $k = b \mod B$. In this section, we use nodes $y_{j,0}, \ldots, y_{j,B-2}$ to approximately recover $x_i^h, x_i^h + 1, \ldots, x_i^{h + B - 1}$ for $h = 0, \ldots, nB - 1$.

In order to measure the error, we employ the Hamming distance between the original vector $x$ and the decoded vector $y$, and denote it by $\text{dist}(x, y)$. The average Hamming distance $\text{dist}(X, Y)$ between the sets of vectors $X = \{x^0, \ldots, x^{n-1}\}$ and $Y = \{y^0, \ldots, y^{n-1}\}$ is defined as

$$\text{dist}(X, Y) = \frac{1}{n} \sum_{i=0}^{n-1} \text{dist}(x^{i}, y^{i}).$$

A. Case of $B = 2$

First, we explain the basic idea using the case of $B = 2$. As in Section IV, we divide the input vectors into $B$ sets.

We assume w.l.o.g. that $(n/2)$ is a positive integer and define $n_2 = (n/2)$. The decoder consists of two layers in addition to the middle layer consisting of nodes $b_i$ ($i = 0, \ldots, d = \lceil \log n \rceil$). We construct nodes $y_i$ ($i = 0, \ldots, n_2 + 1$) for the first layer, and nodes $y_j$ ($j = 0, \ldots, D - 1$) for the second layer (output layer). Recall that $\text{Int}(\beta)$ denotes the integer coded by a binary vector $\beta$. Then, we define activation functions for $y_i$ by

$$y_i = \left[\frac{\text{Int}(\beta)}{2}\right]_1 = i,$$ for $i = 0, \ldots, n_2 - 1$

$$y_{n_2} = [\text{Int}(\beta) = 0 \mod 2)]$$

$$y_{n_2 + 1} = [\text{Int}(\beta) = 1 \mod 2)].$$

For all $j = 0, \ldots, D - 1$, we define weights $w_{ij}$ as in Table I for $i = 0, \ldots, n_2 - 1$, and weights $w_{n_2}$ and $w_{n_2 + 1}$ by

$$w_{n_2} = -1,$$

$$w_{n_2 + 1} = -2.$$

We define the activation function for $y_j$, $j = 0, \ldots, D - 1$, by

$$y_j = \sum_{i=0}^{n_2 + 1} w_{ij}y_i \geq 0.$$

For example, consider the case of $n = 8$ and $D = 3$. Suppose that $x^0_i = (1, 0, 1)$ and $x^1_i = (1, 1, 0)$. Since $x^0_0x^0_1 = 11$, $x^0_0x^1_1 = 10$, and $x^0_1x^1_1 = 10$, we have $w_{0}^0 = 3$, $w_{0}^1 = 2$ and $w_{0}^2 = 1$. For $x^0$, we have $y = (1, 0, 0, 0, 1, 0)$, and thus, $y^0 = (1, 1, 1)$. For $x^1$, we have $y = (1, 0, 0, 0, 0, 1)$ and thus $y^1 = (1, 1, 0)$. Therefore, we can see that the error occurs only on $x^0_i$ for these two samples.

In general, it is seen that the error occurs only when $x^2_i \neq x^2_{i+1} = 1$ and $x^1_i \neq x^1_{i+1} = 1$. Therefore, if all bit patterns are distributed uniformly at random, the expected Hamming distance between $x$ and $y$ is $\left(\frac{D(2^2 - 2)}{D}\right) = \frac{D}{8}$. In order to cope with nonuniform patterns, we can make use of permutations of 0–1 patterns. Details of this trick are described in Sections V-B and V-C.

B. Case of $B = 3$

Next, we consider the case of $B = 3$. We will extend it to a general $B$ in Section V-C.

We assume w.l.o.g. that $(n/3)$ is a positive integer and define $n_3 = (n/3)$. We construct $y_i$ nodes ($i = 0, \ldots, n_3 + 2$), $y_i,0$ nodes ($i = 0, \ldots, D - 1$), and $y_i,1$ nodes ($i = 0, \ldots, D - 1$) as follows (see Fig. 3).

Recall that $\text{Int}(\beta)$ denotes the integer coded by a binary vector $\beta$. Then, the activation function of $y_i$ is defined by

$$y_i = \left[\frac{\text{Int}(\beta)}{3}\right]_1 = i,$$ for $i = 0, \ldots, n_3 - 1$

$$y_{n_3} = [\text{Int}(\beta) = 0 \mod 3)]$$

$$y_{n_3 + 1} = [\text{Int}(\beta) = 1 \mod 3)]$$

$$y_{n_3 + 2} = [\text{Int}(\beta) = 2 \mod 3)].$$

---

**Table I**

| $x^2_i$ | $y^2_i$ | $y^{2+1}_i$ |
|---|---|---|
| 00 | 00 | 00 |
| 01 | 10 | 10 |
| 11 | 11 | 11 |
Fig. 3. Network structure for approximate decoding with $B = 3$, where the middle nodes are omitted.

The activation function for $y_{j,h}$, $j = 0, \ldots, D-1, h = 0, 1$, is

$$y_{j,h} = \left\lfloor \sum_{i=0}^{n_3+2} w^i_{j,h} y_i \right\rfloor.$$

The values of $w^i_{j,h}$, $i = 0, \ldots, n_3 - 1$, $h = 0, 1$ depend on $i$ and $j$ only indirectly through the values of $x^3_j x_j^{3+1} x_j^{3+2}$, i.e., $w^i_{j,h} = w_h(x^3_j x_j^{3+1} x_j^{3+2})$. The values of $w_0$ and $w_1$ are detailed in Table II where binary representation is used for $w_i$; for example, $w_0 = 100(2) = 4$ and $w_1 = 001(2) = 1$ if $x^3_j x_j^{3+1} x_j^{3+2} = 010$.

The corresponding values of the $y_{j,h}$ and the, soon to be defined, $z_j$ are also shown in Table II. The values of $w^i_{j,h}$, $i = n_3, n_3 + 1, n_3 + 2$, $h = 0, 1$ do not depend on $i$ and are

$$w^i_{n_3,0} = -2, \quad w^i_{n_3+1,0} = -4, \quad w^i_{n_3+2,0} = 0,$$

$$w^i_{n_3,1} = -2, \quad w^i_{n_3+1,1} = 0, \quad w^i_{n_3+2,1} = -4.$$

Finally, we construct nodes $z_j$, $j = 0, \ldots, D-1$, with activation functions

$$z_j = \lfloor y_{j,0} + y_{j,1} \rfloor \geq 2.$$

Note that $y_{j,0}^{3+1} = x_j^{3+1}, y_{j,1}^{3+2} = x_j^{3+2}$, and $y_{j,0}^{3} + y_{j,1}^{3} < 2$ for $x^3_j x_j^{3+1} x_j^{3+2} \in \{000, 001, 010\}$, where $y_{j,h}^{3}$ denotes the value of node $y_{j,h}$ for $x^3_j$.

It is seen from Table II that an error occurs only when $x^3_j x_j^{3+1} x_j^{3+2} = 011$. Therefore, if all bit patterns are distributed uniformly at random, the expected Hamming distance between $x'$ and $y'$ is $(D/(3 \cdot 2^3)) = (D/24)$. We can modify the network so that the average Hamming distance between $x'$ and $y'$ is always no more than $(D/24)$.

This can be done as follows. Let $\#_{b_0 b_1 b_2} = \sum_{j=0}^{D-1} [i(j x^3_j x_j^{3+1} x_j^{3+2} = b_0 b_1 b_2)]$ where $b_0 b_1 b_2 \in \{0, 1\}$. Let $c_D c_{D+1} = \arg\min_{b_0 b_1 b_2} \#_{b_0 b_1 b_2}$, where the tie can be broken in any way. Then, we let $d \Delta a D_{a_2} = (0 \oplus c_3)(1 \oplus c_1)(1 \oplus c_2)$. For example, $a_D a_{D_2} = 011$ when $c_D c_{D+1} = 000$.

Here, we define $\chi_{\Delta a D_{a_2}}(x_0 x_1 x_2)$ by

$$\chi_{\Delta a D_{a_2}}(x_0 x_1 x_2) = (x_0 \oplus a_0)(x_1 \oplus a_1)(x_2 \oplus a_2).$$

For example, $\chi_{011}(000) = 011$, $\chi_{011}(001) = 010$, and $\chi_{011}(011) = 000$ when $a_D a_1 a_2 = 011$. Then, we define $w^i_{j,h}$ by

$$w^i_{j,0} = w_0 \left( \chi_{\Delta a D_{a_2}}(x^3_j x_j^{3+1} x_j^{3+2}) \right),$$

$$w^i_{j,1} = w_1 \left( \chi_{\Delta a D_{a_2}}(x^3_j x_j^{3+1} x_j^{3+2}) \right).$$

Finally, we define nodes $z_j'$ in the output layer by

$$z_j' = z_j \oplus \left((a_0 y_{n_3}) \lor (a_1 y_{n_3+1}) \lor (a_2 y_{n_3+2})\right)$$

which can be realized by adding nodes $y'$, $y''$, and $z_{j,h}$ ($j = 0, \ldots, D-1, h = 0, 1$) and by defining the activation functions

$$y' = [a_0 y_{n_3} + a_1 y_{n_3+1} + a_2 y_{n_3+2} \geq 1],$$

$$y'' = [y' \geq 1],$$

$$z_{i,0} = [z_i - y' \geq 1],$$

$$z_{i,1} = [-z_i + y'' \geq 1],$$

$$z_j' = [z_j + z_{i,1} \geq 1].$$

For example, when $c_D c_{D+1} = 000$, we have the values of relevant variables as in Table III.

Indeed, if $x^3_j x_j^{3+1} x_j^{3+2} = 010$, we have $z_j' x_j^{3+1} x_j^{3+2} = 001$ and $(z_j') x_j^{3+1} (z_j') x_j^{3+2} = 010$ by the following:

$$w^i_{j,0} = w_0(\chi_{011}(010)) = w_0(001) = 001(2) = 1,$$

$$w^i_{j,1} = w_1(\chi_{011}(010)) = w_1(001) = 100(2) = 4$$

for $3i$.

$$y_{j,0} = [1 - 2 \geq 0] = 0,$$

$$y_{j,1} = [4 - 2 \geq 0] = 1,$$

$$z_j = [0 + 1 \geq 2] = 0,$$

$$z_j' = 0 \oplus (0 \cdot y_{n_3}) = 0$$

for $3i + 1$.

$$y_{j,0} = [1 - 4 \geq 0] = 0,$$

$$y_{j,1} = [4 - 4 \geq 0] = 0.$$
is given by \( \max \) to-one to

\[ C. \text{ General Case} \]

that and width \( \max \), . . . , \( n \)

\[ h \]

\[ B \]

\[ j \]

\[ z_j = [0 + 1 \geq 2] = 0 \]

\[ z'_j = 0 \oplus (1 \cdot \gamma_{n+1}) = 1 \]

[for \( 3i + 2 \)]

\( y_{j,0} = [1 - 0 \geq 0] = 1 \)

\( y_{j,1} = [4 - 4 \geq 0] = 1 \)

\[ z_j = [1 + 1 \geq 2] = 1 \]

\[ z'_j = 1 \oplus (1 \cdot \gamma_{n+2}) = 0. \]

If \( n \) can be divided by 3, the total error will be at most \((nD/(3 \cdot 2^3)) = (nD/24)\). Otherwise, the total error will be at most \((nD/(3 \cdot 2^3)) + D = (nD/24) + D\) because there might be an additional error per \( j \). Since the maximum width is given by \( \max(\lceil n/3 \rceil + 3, 2D + 1) \), we have the following theorem.

**Theorem 9:** For any perfect encoder that maps \( X_n \) one-to-one to \( d \)-dimensional binary vectors, with \( d = \lceil \log n \rceil \) and \( n \) sufficiently large, there exists a decoder of a constant depth and width \( \max(\lceil n/3 \rceil + 3, 2D + 1) \) whose average Hamming distance error is at most \( D((1/24) + (1/n)) \).

\[ \begin{array}{c|c|c|c}
\hline
x_j & z_j & \gamma_{011} & z'_j \\
\hline
000 & 011 & 111 & 100 \\
001 & 010 & 010 & 001 \\
010 & 001 & 001 & 010 \\
011 & 000 & 000 & 011 \\
100 & 111 & 111 & 100 \\
101 & 110 & 110 & 101 \\
110 & 101 & 101 & 110 \\
111 & 100 & 100 & 111 \\
\hline
\end{array} \]

\[ \text{TABLE III} \]

VALUES OF RELEVANT VARIABLES FOR THE CASE OF \( c_{01}c_2 = 000 \)

For integer \( k \in \{0, \ldots, 2^B - 1\} \), let \( k_0, k_1, \ldots, k_{B-1} \) be the binary representation of \( k \) [i.e., \( k = \text{Int}(k_0, k_1, \ldots, k_{B-1}) \)], where \( k_0 \) is the most significant bit. We define \( w_h(k) (h = 0, \ldots, B - 2) \) by

\[ w_0(k_0k_1k_2k_3, \ldots, k_{B-1}) = \text{Int}(k_0k_1k_2k_3, \ldots, k_{B-1}) \]

\[ w_1(k_0k_1k_2k_3, \ldots, k_{B-1}) = \text{Int}(k_0k_1k_2k_3, \ldots, k_{B-1}) \]

\[ w_2(k_0k_1k_2k_3, \ldots, k_{B-1}) = \text{Int}(k_0k_1k_2, \ldots, k_{B-1}) \]

\[ \ldots \]

\[ w_{B-2}(k_0k_1k_2k_3, \ldots, k_{B-1}) = \text{Int}(k_0k_1, \ldots, k_{B-1}). \]

Then, we define \( w_{i,h}^j = w_h(x_i^j, x_{i+1}^j, \ldots, x_{i+B-1}^j) \) for \( j = 0, \ldots, D - 1, i = 0, \ldots, n_B - 1 \), and \( h = 0, \ldots, B - 2 \). The following proposition is straightforward from the definitions of \( y_{i,h}, s_i, w_{n_B+h}, s_i \), and \( z_i, s_i \).

**Proposition 10:** If \( x_i^j, x_{i+1}^j, \ldots, x_{i+B-1}^j \) is \( 011, \ldots, 1 \), then \( z_i^j, z_i^{j+1}, \ldots, z_i^{j+B-1} = 111, \ldots, 1 \) holds. Otherwise, \( z_i^j, z_i^{j+1}, \ldots, z_i^{j+B-1} = x_i^j, x_{i+1}^j, \ldots, x_{i+B-1}^j \) holds.

As in Section V-B, it is seen from this proposition that if all bit patterns are distributed uniformly at random, the expected Hamming distance between \( x' \) and \( y' \) is \((D/B^2)B\). We can modify the network so that the average Hamming distance between \( x' \) and \( y' \) is always no more than \((D/B^2)B\) as well. The modification method is given in the following, which is almost the same as that in Section V-B.

Let \( \#b_1, \ldots, b_{B-1} = \sum_{j=0}^{B-1} \{[i]x_i^j, x_{i+1}^j, \ldots, x_{i+B-1}^j = b_1, \ldots, b_{B-1} \} \) where \( b_0, b_1, \ldots, b_{B-1} \in \{0, 1\}^B \). Let \( c_0c_1, c_{B-1} \) be \( \arg \min_{b_1, \ldots, b_{B-1}} \#ac_{c_0b_1, \ldots, b_{B-1}} \) where the tie can be broken in any way. Then, we let \( a_0a_1a_2a_3 = (0 \oplus c_0)(1 \oplus c_1), \ldots, (1 \oplus c_{B-1}) \). For example. \( a_0a_1a_2a_3 = 0110 \) when \( c_0c_1c_2c_3 = 0001 \).

Here, we define \( x_{a_0a_1 \cdots, a_{B-1}}(x_0x_1, \ldots, x_{B-1}) \) by

\[ x_{a_0a_1 \cdots, a_{B-1}}(x_0x_1, \ldots, x_{B-1}) = (x_0 \oplus a_0)(x_1 \oplus a_1), \ldots, (x_{B-1} \oplus a_{B-1}) \]

Then, we define \( w_{i,h}^j \) by

\[ w_{i,h}^j = w_h(x_{a_0a_1 \cdots, a_{B-1}}(x_i^j, x_{i+1}^j, \ldots, x_{i+B-1}^j)) \]

for \( j = 0, \ldots, D - 1, i = 0, \ldots, n_B - 1 \), and \( h = 0, \ldots, B - 2 \).

Finally, we define nodes \( z'_j \) in the output layer by

\[ z'_j = z_j \oplus \bigoplus_{h=0}^{B-1} a_hy_{n_B+h} \]

which can be realized by adding nodes \( y', y'' \), and \( z_{j,h} (j = 0, \ldots, D - 1, h = 0, 1) \) and defining the.
Since the maximum width is given by \(\max([n/B] + B, (B - 1)D + 1)\), we have the following theorem.

**Theorem 11:** For any perfect encoder that maps \(X_n\) one-to-one to \(d\)-dimensional binary vectors, with \(d = \lceil \log n \rceil\) and sufficiently large \(n\), there exists a decoder of a constant depth and width \(\max([n/B] + B, (B - 1)D + 1)\) whose average Hamming distance error is at most \(D((1/B2^d)+1/n)\), where \(B\) is any integer such that \(2 \leq B \leq n\).

Note that this theorem is a generalized version of Theorem 9 and the resulting width is smaller than that of Theorem 7 for all \(B \geq 2\) and \([n/B] + B > BD\) (i.e., \(D < n'\) for \(n' \approx n/B^2\)) although the corresponding decoder is an approximate one and uses more layers. Note also that if \(B = 2^k\), the architecture of the decoder can be explicitly described as \(d/[n/B] + B/(B - 1)D + 1/D + 1/2D/D\) where \(d = \lceil \log n \rceil\). Furthermore, if we can consider the average case error over binary input vectors given uniformly at random, this architecture can be reduced to \(d/[n/B] + B/(B - 1)D/D\).

**VI. Conclusion**

In this brief paper, we showed an improved upper bound on the size/width of the decoder in a Boolean threshold autoencoder. We also showed lower bounds on the size of the decoder. Although upper and lower bounds are relatively close, there still exists a gap. Therefore, closing the gap is left as future work. It is worth noting that, although we did not consider here the learning of parameters such as weights and thresholds, our lower bounds hold also in such a setting, i.e., no learning method can find parameters with which the input vectors are perfectly recovered if the size of the decoder is less than the lower bound. On the other hand, since our upper bounds were obtained by explicit construction, it is possible that learning methods can improve on these bounds. Thus, the development of a learning method that can determine parameters with which the input vectors are perfectly recovered when using autoencoders of the appropriate size may be interesting future work.

Finally, it is to be noted that this brief paper focused on a specific aspect of Boolean threshold autoencoders. Since Boolean threshold autoencoders are not widely used, the compressive power and its learnability of other types of autoencoders should be studied.

**References**

[1] M. Anthony, *Discrete Mathematics of Neural Networks*. Philadelphia, PA, USA: SIAM, 2001.

[2] J. Šíma and P. Orponen, “General-purpose computation with neural networks: A survey of complexity theoretic results,” *Neural Comput.*, vol. 15, no. 12, pp. 2727–2778, Dec. 2003.

[3] K.-Y. Siu, V. Rndoychowdhury, and T. Kailath, *Discrete Neural Computation: A Theoretical Foundation*. Upper Saddle River, NJ, USA: Prentice-Hall, 1995.

[4] C. Doesch, “Tutorial on variational autoencoders,” 2016, arXiv:1606.05908.

[5] M. Tschannen, O. Bachem, and M. Lucic, “Recent advances in autoencoder-based representation learning,” 2018, arXiv:1812.05609.

[6] R. Gómez-Bombarelli et al., “Automatic chemical design using a data-driven continuous representation of molecules,” *ACS Central Sci.*, vol. 4, no. 2, pp. 268–276, Feb. 2018.

[7] P. Li, Y. Pei, and J. Li, “A comprehensive survey on design and application of autoencoder in deep learning,” *Appl. Soft Comput.*, vol. 138, May 2023, Art. no. 110176.

[8] D. Ackley, G. Hinton, and T. Sejnowski, “A learning algorithm for Boltzmann machines,” *Cognit. Sci.*, vol. 9, no. 1, pp. 147–169, Mar. 1985.

[9] P. Baldi and K. Hornik, “Neural networks and principal component analysis: Learning from examples without local minima,” *Neural Netw.*, vol. 2, no. 1, pp. 53–58, 1989.

[10] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[11] P. Baldi, “Autoencoders, unsupervised learning, and deep architectures,” in *Proc. Workshop Conf.*, vol. 27, 2012, pp. 37–50.

[12] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, “Energy compaction-based image compression using convolutional autoencoder,” *IEEE Trans. Multimedia*, vol. 22, no. 4, pp. 860–873, Apr. 2020.

[13] A. Grover and S. Ermon, “Uncertainty autoencoders: Learning compressed representations via variational information maximization,” in *JMLR Proc. Workshop 22nd Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2019, pp. 2514–2524.

[14] C. Jia, Z. Liu, Y. Wang, S. Ma, and W. Gao, “Layered image compression using scalable auto-encoder,” in *Proc. IEEE Conf. Multimedia Inf. Process. Retr. (MIPR)*, Mar. 2019, pp. 431–436.

[15] T. Kärkkäinen and J. Hänninen, “Additive autoencoder for dimension estimation,” *Neurocomputing*, vol. 551, Sep. 2023, Art. no. 126520.

[16] S. Yu and J. C. Principe, “Understanding autoencoders with information theoretic concepts,” *Neural Netw.*, vol. 117, pp. 104–123, Sep. 2019.

[17] N. I. Tapia and P. A. Estevez, “On the information plane of autoencoders,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.

[18] S. Lee and J. Jo, “Information flows of diverse autoencoders,” *Entropy*, vol. 23, no. 7, p. 862, Jul. 2021.

[19] G. Montufar, R. Pascanu, K. Cho, and Y. Bengio, “On the number of linear regions of deep neural networks,” in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 27, 2014, pp. 2924–2932.

[20] S. An, M. Bennamoun, and F. Boussaid, “On the compressive power of deep rectifier networks for high resolution representation of class boundaries,” 2017, arXiv:1708.07244v1.

[21] C. Yun, S. Sra, and A. Jadbabaie, “Small ReLU networks are powerful memorizers: A tight analysis of memorization capacity,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 15532–15543.

[22] R. Vershynin, “Memory capacity of neural networks with threshold and rectified linear unit activations,” *SIAM J. Math. Data Sci.*, vol. 2, no. 4, pp. 1004–1033, Jan. 2020.

[23] A. A. Melkman, S. Guo, W.-K. Ching, P. Liu, and T. Akutsu, “On the compressive power of Boolean threshold autoencoders,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 2, pp. 921–931, Feb. 2023.

[24] K.-Y. Siu, J. Bruck, T. Kailath, and T. Hofmeister, “Depth efficient neural networks for division and related problems,” *IEEE Trans. Inf. Theory*, vol. 39, no. 3, pp. 946–956, May 1993.