Provable Memorization via Deep Neural Networks using Sub-linear Parameters

Sejun Park   SEJUN.PARK@VECTORINSTITUTE.AI
Vector Institute for Artificial Intelligence

Jaeho Lee   JAEHO-LEE@KAIST.AC.KR
Korea Advanced Institute of Science and Technology

Chulhee Yun   CHULHEEY@MIT.EDU
Massachusetts Institute of Technology

Jinwoo Shin   JINWOOS@KAIST.AC.KR
Korea Advanced Institute of Science and Technology

Editors: Mikhail Belkin and Samory Kpotufe

Abstract

It is known that $O(N)$ parameters are sufficient for neural networks to memorize arbitrary N input-label pairs. By exploiting depth, we show that $O(N^{2/3})$ parameters suffice to memorize N pairs, under a mild condition on the separation of input points. In particular, deeper networks (even with width 3) are shown to memorize more pairs than shallow networks, which also agrees with the recent line of works on the benefits of depth for function approximation. We also provide empirical results that support our theoretical findings.

1. Introduction

The modern trend of over-parameterizing neural networks has shifted the focus of deep learning theory from analyzing their expressive power toward understanding the generalization capabilities of neural networks. While the celebrated universal approximation theorems state that over-parameterization enables us to approximate the target function with a smaller error (Cybenko, 1989; Pinkus, 1999), the theoretical gain is too small to satisfactorily explain the observed benefits of over-parameterizing already-big networks. Instead of "how well can models fit," the question of "why models do not overfit" has become the central issue (Zhang et al., 2017).

Ironically, a recent breakthrough on the phenomenon known as the *double descent* (Belkin et al., 2019; Nakkiran et al., 2020) suggests that answering the question of "how well can models fit" is in fact an essential element in fully characterizing their generalization capabilities. In particular, the double descent phenomenon characterizes two different phases according to the capability/incapability of the network size for memorizing training samples. If the network size is insufficient for memorization, the traditional bias-variance trade-off occurs. However, after the network reaches the capacity that memorizes the dataset, i.e., "interpolation threshold," larger networks exhibit better generalization. Under this new paradigm, identifying the *minimum size of networks* for memorizing *finite* input-label pairs becomes a key issue, rather than function approximation that considers *infinite* inputs.

The memory capacity of neural networks is relatively old literature, where researchers have studied the minimum number of parameters for memorizing arbitrary N input-label pairs. Existing results showed that $O(N)$ parameters (i.e., weights and biases) are sufficient for various activation

functions (Baum, 1988; Huang and Babri, 1998; Huang, 2003; Yun et al., 2019; Vershynin, 2020). On the other hand, Sontag (1997) established the negative result that for any network using analytic definable activation functions with $o(N)$ parameters, there exists a set of N input-label pairs that the network cannot memorize. The sub-linear number of parameters also appear in a related topic, namely the VC-dimension of neural networks. It has been proved that there exists a set of N inputs such that a neural network with $o(N)$ parameters can "shatter," i.e., memorize arbitrary labels (Maass, 1997; Bartlett et al., 2019). Comparing the two results on $o(N)$ parameters, Sontag (1997) showed that *not all* sets of N inputs can be memorized for arbitrary labels, whereas Bartlett et al. (2019) showed that *at least one* set of N inputs can be shattered. This suggests that there may be a reasonably large family of N input-label pairs that can be memorized with $o(N)$ parameters, which is our main interest.

1.1. Summary of results

In this paper, we identify a mild condition satisfied by many practical datasets, and show that $o(N)$ parameters suffice for memorizing such datasets. To bypass the negative result by Sontag (1997), we introduce a condition to the set of inputs, called the Δ-separateness.

Definition 1 *For $\mathcal{X} \subset \mathbb{R}^{d_x}$, we say $\mathcal{X}$ is Δ-separated if*

$$\sup_{x,x' \in \mathcal{X}: x \neq x'} \|x - x'\|_2 < \Delta \times \inf_{x,x' \in \mathcal{X}: x \neq x'} \|x - x'\|_2.$$

This condition requires that the ratio of the maximum distance to the minimum distance between distinct points is bounded by Δ. Here, the condition is milder when Δ is bigger. Notice that any given finite set of (distinct) inputs is Δ-separated for some Δ, so one might ask why Δ-separateness is different from having distinct inputs in a dataset. The key difference is that even if the number of data points N grows, the ratio of the maximum to the minimum should remain bounded by some Δ. Given the discrete nature of computers, many practical datasets satisfy Δ-separateness, as we will see shortly. Also, this condition is more general than the minimum distance assumptions ($\forall i, \|x_i\|_2 = 1, \forall i \neq j, \|x_i - x_j\|_2 \geq \rho > 0$) that are employed in existing theoretical results (Hardt and Ma, 2017; Vershynin, 2020). To see this, note that the minimum distance assumption implies $2/\rho$-separateness. In our theorem statements, we will use the phrase "Δ-separated set of N pairs" to refer to N input-label pairs, where the set of inputs is Δ-separated.

In our main theorem sketched below, we prove the sufficiency of $o(N)$ parameters for memorizing any Δ-separated set of N pairs (i.e., any Δ-separated set of N inputs with arbitrary labels) even for large Δ. More concretely, our result is of the following form:

Theorem 1 (Informal) *For any $w \in [2/3, 1]$, there exists a $O(N^{2-2w} + \log \Delta)$-layer, $O(N^w + \log \Delta)$-parameter fully-connected network using a sigmoidal or* RELU *activation function that can memorize any Δ-separated set of N pairs.*

Theorem 1 implies that for $w = 2/3$ and $\Delta = 2^{O(N^{2/3})}$, $O(N^{2/3})$ parameters are sufficient for memorizing any Δ-separated set of N pairs. Here, we can check from Definition 1 that the $\log \Delta$ term does not usually dominate the depth or the number of parameters, especially for modern deep architectures and practical datasets. For example, it is easy to check that any dataset consisting of 3-channel images (values from $\{0, 1, \ldots, 255\}$) of size $a \times b$ satisfies $\log \Delta < (9 + \frac{1}{2}\log(ab))$ (e.g.,

$\log \Delta < 17$ for the ImageNet dataset), which is often much smaller than the depth of modern deep architectures.

For practical datasets, we can show that networks with parameters fewer than the number of pairs can successfully memorize the dataset. For example, in order to perfectly classify one million images in the ImageNet dataset[1] with 1000 classes, our result shows that 0.7 million parameters are sufficient. The improvement is more significant for larger datasets. To memorize 15.8 million bounding boxes in the Open Images V6 dataset[2] with 600 classes, our result shows that only 4.5 million parameters suffice.

For a large class (i.e., $2^{O(N^w)}$-separated) of datasets with N pairs, Theorem 1 improves the number of parameters suffcent for memorization from $O(N)$ down to $O(N^w)$ for any $w \in [2/3, 1]$, by exploiting network depth that increases with the number of pairs N. Then, it is natural to ask whether the depth increasing with N is *necessary* for memorization with a sub-linear number of parameters. The following existing result on the VC-dimension implies that increasing depth is necessary for memorization with $o(N/\log N)$ parameters, at least for RELU networks.

Theorem *[Bartlett et al. (2019)]* **(Informal)** *For L-layer RELU networks, $\Omega(N/(L\log N))$ parameters are necessary for memorizing at least one set of N inputs with arbitrary labels.*

The above theorem implies that for RELU networks of constant depth, $\Omega(N/\log N)$ parameters are necessary for memorizing at least one set of N inputs with arbitrary labels. In contrast, by increasing depth with N, Theorem 1 shows that there is a large class of datasets that can be memorized with $o(N/\log N)$ parameters. Combining these two results, one can conclude that increasing depth is *necessary and sufficient* for memorizing a large class of N pairs with $o(N/\log N)$ parameters.

Given that the depth is critical for memorization with $o(N/\log N)$ parameters, is the width also critical? We prove that it is not the case, via the following theorem.

Theorem 2 (Informal) *For a fully-connected network of width 3 using a sigmoidal or RELU activation function, $O(N^{2/3} + \log \Delta)$ parameters (i.e., layers) suffice to memorize any Δ-separated set of N pairs.*

Theorem 2 states that under $2^{O(N^{2/3})}$-separateness of inputs, the network width does not necessarily have to increase with N for memorization with sub-linear parameters. Furthermore, it shows that even a surprisingly narrow network of width 3 has a superior memorization power than a fixed-depth network, requiring only $O(N^{2/3})$ parameters for memorizing any $2^{O(N^{2/3})}$-separated N pairs.

Theorems 1 and 2 show that we can construct some *tailor-made* network architectures that can memorize N pairs with $o(N)$ parameters, under the Δ-separateness condition. This means that these theorems do not answer the question of how many such data points can a *given* network memorize. To answer this question, we provide sufficient conditions for identifying the maximum number of points given general networks (Theorem 3). In a nutshell, our conditions indicate that to memorize more pairs under the same budget for the number of parameters, the network must be deep, and most of its parameters should be concentrated in the first layers. In other words, the network must have a deep and narrow architectures in the final layers, as also proposed in (Bartlett et al., 2019). Our sufficient conditions successfully incorporate the characteristics of datasets, the number of parameters, and the architecture, which enable us to memorize Δ-separated datasets with

---

1. http://www.image-net.org/
2. https://storage.googleapis.com/openimages/web/index.html

number of pairs super-linear in the number of parameters. This is in contrast to the prior results that the number of arbitrary pairs that can be memorized is at most proportional to the number of parameters (Yamasaki, 1993; Yun et al., 2019; Vershynin, 2020).

Finally, we provide empirical results corroborating our theoretical findings that deep networks often memorize better than their shallow counterparts with a similar number of parameters. Here, we emphasize that better memorization power does not necessarily imply better generalization. We indeed observe that shallow and wide networks often generalize better than deep and narrow networks, given the same (or similar) training accuracy.

Organization. We first introduce related works in Section 2. In Section 3, we introduce notation and the problem setup. We formally state our main results and discuss them in Section 4. We provide the proof of our main theorem in Section 5. In Section 6, we provide empirical observations on the effect of depth and width in neural networks. Finally, we conclude the paper in Section 7.

2. Related works

While introducing related works, we hide the input dimension d_x in O, Ω, Θ as in convention. We note that in our main results in Section 4, O, Ω, Θ will only hide universal constants, excluding d_x.

Sufficient number of parameters for memorization. Identifying the sufficient number of parameters for memorizing arbitrary N pairs has a long history. Earlier works mostly focused on bounding the number of hidden neurons in shallow networks required for memorization. Baum (1988) proved that for 2-layer STEP[3] networks, $O(N)$ hidden neurons (i.e., $O(N)$ parameters) are sufficient for memorizing arbitrary N pairs when inputs are in general position. Huang and Babri (1998) showed that the same bound holds for any bounded and nonlinear activation function σ satisfying that either $\lim_{x \to -\infty} \sigma(x)$ or $\lim_{x \to \infty} \sigma(x)$ exists, without any condition on inputs. The $O(N)$ bounds on the number of hidden neurons was improved to $O(\sqrt{N})$ by exploiting an additional hidden layer by Huang (2003); nevertheless, this construction still requires $O(N)$ parameters.

With the advent of deep learning, the focus has shifted to modern activation functions and deeper architectures. Zhang et al. (2017) proved that $O(N)$ hidden neurons are sufficient for 2-layer RELU networks to memorize arbitrary N pairs. Yun et al. (2019) showed that for deep RELU (or hard $\tanh$) networks having at least 3 layers, $O(N)$ parameters are sufficient. Vershynin (2020) proved a similar result for STEP (or RELU) networks for memorizing arbitrary set of unit vectors $\{x_i\}_{i=1}^N$ satisfying $\|x_i - x_j\|_2^2 = \Omega(\frac{\log \log d_{\max}}{\log d_{\min}})$, $N = e^{O(d_{\min}^{1/5})}$, and $d_{\max} = e^{O(d_{\min}^{1/5})}$ where $d_{\max}$ and $d_{\min}$ denote the maximum and the minimum hidden dimensions, respectively.

In addition, the memorization power of modern network architectures has also been studied. Hardt and Ma (2017) showed that RELU networks consisting of residual blocks with $O(N)$ hidden neurons can memorize any arbitrary set of unit vectors $\{x_i\}_{i=1}^N$ satisfying $\|x_i - x_j\|_2 \geq \rho$ for some absolute constant $\rho > 0$. Nguyen and Hein (2018) studied a broader class of layers and proved that $O(N)$ hidden neurons suffice for convolutional neural networks consisting of fully-connected, convolutional, and max-pooling layers for memorizing arbitrary N pairs having different patches.

Necessary number of parameters for memorization. The necessary number of parameters for memorization has also been studied. Sontag (1997) showed that for any neural network using analytic definable activation functions, $\Omega(N)$ parameters are necessary for memorizing arbitrary N

---

3. STEP denotes the binary threshold activation function: $x \mapsto \mathbf{1}[x \geq 0]$.

pairs. Namely, given any network using analytic definable activation with $o(N)$ parameters, there exists a set of N pairs that the network cannot memorize.

The Vapnik-Chervonenkis (VC) dimension is also closely related to the memorization power of neural networks. While the memorization power studies the number of parameters for memorizing *arbitrary N* pairs, the VC-dimension is related to the number of parameters for memorizing *at least one set* of N inputs with arbitrary labels. In particular, upper bounds on the VC-dimension translates to lower bounds on the necessary number of parameters for memorizing arbitrary N pairs. The VC-dimension of neural networks has been studied for various types of activation functions. For memorizing at least one set of N inputs with arbitrary labels, it is known that $\Theta(N/\log N)$ parameters are necessary (Baum and Haussler, 1989) and sufficient (Maass, 1997) for STEP networks. Similarly, Karpinski and Macintyre (1997) proved that $\Omega(\sqrt{N}/U)$ parameters are necessary for sigmoid networks of U neurons. Recently, Bartlett et al. (2019) showed that $\Theta(N/(\bar{L}\log N))$ parameters are necessary and sufficient for L-layer networks using any piecewise linear activation function where $\bar{L} := \frac{1}{W_L}\sum_{\ell=1}^{L} W_\ell$ and W_ℓ denotes the number of parameters up to the ℓ-th layer.

Benefits of depth in neural networks   To understand deep learning, researchers have investigated the advantages of deep neural networks compared to shallow neural networks with a similar number of parameters. Initial results discovered examples of deep neural networks that cannot be approximated by shallow neural networks without using exponentially many parameters (Telgarsky, 2016; Eldan and Shamir, 2016; Arora et al., 2018). Recently, it is discovered that deep neural networks require fewer parameters than shallow neural networks to represent or approximate a class of periodic functions (Chatziafratis et al., 2020a,b). For approximating continuous functions, Yarotsky (2018) proved that the number of required parameters for RELU networks of constantly bounded depth are square to that for deep RELU networks.

3. Problem setup

In this section, we describe the problem setup and notation. First, we introduce frequently used notation. We use $\log$ to denote the logarithm to the base 2. We let RELU be the function $x \mapsto \max\{x, 0\}$, STEP be the function $x \mapsto \mathbf{1}[x \geq 0]$, and ID be the function $x \mapsto x$. For $\mathcal{X} \subset \mathbb{R}$, we denote $\lfloor\mathcal{X}\rfloor := \{\lfloor x \rfloor : x \in \mathcal{X}\}$. For $n \in \mathbb{N}$ and a set $\mathcal{X}$, we denote $\binom{\mathcal{X}}{n} := \{\mathcal{S} \subset \mathcal{X} : |\mathcal{S}| = n\}$. For $n \in \mathbb{N}$, we use $[n] := \{0, \ldots, n-1\}$. We also extend the modulo operation to $x \bmod y := x - y \cdot \lfloor\frac{x}{y}\rfloor$ for $x \geq 0$ and $y > 0$. To describe our results precisely, we use $O(\cdot), \Theta(\cdot), \Omega(\cdot)$ in our theorems and lemmas only to hide universal constants, excluding the input dimension d_x.

Throughout this paper, we consider fully-connected feedforward networks. In particular, we consider the following setup: Given an activation function σ, we define a neural network f_θ of L layers (or equivalently $L-1$ hidden layers), input dimension d_x, output dimension 1, and hidden layer dimensions $d_1, \ldots, d_{L-1}$ parameterized by θ as $f_\theta := t_{L-1} \circ \sigma \circ \cdots \circ t_1 \circ \sigma \circ t_0$. Here, $t_\ell : \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_{\ell+1}}$ is an affine transformation parameterized by θ.[4] We count the number of parameters as the number of weights and biases in networks. We define the width of f_θ as the maximum over $d_1, ..., d_{L-1}$. We denote a neural network using an activation function σ by a "σ network" and a neural network using two activation functions σ_1, σ_2 by a "$\sigma_1 + \sigma_2$ network."

As we introduced in Section 1.1, our main results hold for any sigmoidal activation function and RELU. Formally, we define the *sigmoidal* functions as follows.

---

4. We set $d_0 := d_x$ and $d_L := 1$.

Definition 2 *We say a function $\sigma : \mathbb{R} \to \mathbb{R}$ is sigmoidal if the following conditions hold.*

- *Both $\lim_{x \to -\infty} \sigma(x)$, $\lim_{x \to \infty} \sigma(x)$ exist and $\lim_{x \to -\infty} \sigma(x) \neq \lim_{x \to \infty} \sigma(x)$.*
- *There exists $z \in \mathbb{R}$ such that σ is continuously differentiable at z and $\sigma'(z) \neq 0$.*

A class of sigmoidal functions covers many activation functions including sigmoid, $\tanh$, hard $\tanh$, etc. Furthermore, since hard $\tanh$ can be represented as a composition or a summation of two RELU functions,[5] our results for sigmoidal activation functions hold for RELU as well, with an additional constant factor on depth or width.

Lastly, we formally define the memorization as follows.

Definition 3 *Given $C, d_x \in \mathbb{N}$, a set of inputs $\mathcal{X} \subset \mathbb{R}^{d_x}$, a label function $y : \mathcal{X} \to [C]$, and a network f_θ parameterized by θ, we say f_θ can memorize $\{(x, y(x))\}_{x \in \mathcal{X}}$ in d_x dimension with C classes if for any $\varepsilon > 0$, there exists θ such that $|f_\theta(x) - y(x)| \leq \varepsilon$ for all $x \in \mathcal{X}$.*

In Definition 3, we define memorizability as the ability to uniformly *approximate* the labels up to an arbitrarily small error $\varepsilon > 0$, instead of the exact memorization (i.e., $\varepsilon = 0$). This *almost lossless* definition allows us to use the following two-step proof strategy: (1) Construct a STEP + ID network that exactly memorizes the dataset, (2) Approximate the STEP + ID network by another network using *any* sigmoidal activation function; see Section 5 for a more detailed outline. Nevertheless, the exact memorization holds for the activation functions where an exact implementation of STEP + ID network for finite inputs is possible, e.g., hard $\tanh$ or RELU. We often write "f_θ can memorize arbitrary N pairs" without "in d_x dimension with C classes" throughout the paper.

4. Main results

4.1. Memorization via sub-linear parameters

Efficacy of depth for memorization. Now, we introduce our main theorem on memorizing N pairs with $o(N)$ parameters. The proof of Theorem 1 is presented in Section 5.

Theorem 1 *For any $C, N, d_x \in \mathbb{N}$, $w \in [2/3, 1]$, and a sigmoidal activation function σ, there exists a σ network f_θ of $O\left(\log d_x + \log \Delta + \frac{N^{2-2w}}{1+(1.5w-1)\log N} \log C\right)$ hidden layers and $O\left(d_x + \log \Delta + N^w + N^{1-w/2} \log C\right)$ parameters such that f_θ can memorize any Δ-separated set of N pairs in d_x dimension with C classes.*

Note that the $1 + (1.5w - 1) \log N$ denominator exists in the number of layers in Theorem 1, which is omitted in its informal version in Section 1.1. In addition, while we only address sigmoidal activation functions in the statement of Theorem 1, the same conclusion holds for RELU networks as we described in Section 3.

In Theorem 1, Δ induces only $O(\log \Delta)$ overhead to the number of layers and the number of parameters. As we introduced in Section 1.1, $\log \Delta$ for modern datasets is often very small. Furthermore, $\log \Delta$ can be small for random inputs. For example, a set of d_x-dimensional i.i.d. standard normal random vectors of size N satisfies $\log \Delta = O(\frac{1}{d_x} \log(N/\sqrt{\delta}))$ with probability at least $1 - \delta$ (see Lemma 9 in Appendix C). Hence, the Δ-separateness condition is often negligible.

Suppose that d_x and C are treated as constants, as also assumed in existing results. Then, Theorem 1 implies that if $\log \Delta = O(N^w)$ for some $w < 1$, then $\Theta(N^w)$ (i.e., sub-linear to N)

---

5. hard $\tanh(x) = \text{RELU}(x + 1) - \text{RELU}(x - 1) - 1 = \text{RELU}\big(2 - \text{RELU}(1 - x)\big) - 1$

parameters are *sufficient* for networks using a sigmoidal or RELU activation function to memorize arbitrary Δ-separated set of N pairs. Note that the condition $\log \Delta = O(N^w)$ is very loose for many practical datasets, especially for those with huge N. Combined with the lower bound $\Omega(N/\log N)$ on the *necessary* number of parameters for RELU networks of constant depth (Bartlett et al., 2019), Theorem 1 implies that the depth growing in N is *necessary and sufficient* for memorizing a large class (i.e., Δ-separated) of N pairs with $o(N/\log N)$ parameters. In other words, deep RELU networks have stronger memorization power than shallow RELU networks.

Unimportance of width for memorization. Given that the depth is critical for memorization with $o(N/\log N)$ parameters, we show that the width is not very critical. Specifically, we prove that extremely narrow networks of width 3 can memorize with $O(N^{2/3})$ layers (i.e., $O(N^{2/3})$ parameters) under $\log \Delta = O(N^{2/3})$ and constant d_x, as stated in the following theorem. The proof of Theorem 2 is presented in Section 4.2.

Theorem 2 *For any $C, N, d_x \in \mathbb{N}$, $\Delta \geq 1$, and sigmoidal activation function σ, a σ network of $\Theta(\log d_x + \log \Delta + N^{2/3} \log C)$ hidden layers and width 3 can memorize any Δ-separated set of N pairs in d_x dimension with C classes.*

The statement of Theorem 2 might be somewhat surprising since the network width for memorization does not depend on the input dimension d_x. This is in contrast with the recent universal approximation results that width at least $d_x + 1$ is necessary for approximating functions on d_x-dimensional domains (Lu et al., 2017; Hanin and Sellke, 2017; Johnson, 2019; Park et al., 2021). The main difference follows from the fundamental difference in the two approximation problems, i.e., approximating a function at finite inputs versus infinite inputs (e.g., the unit cube). Any set of N distinct input vectors can be easily mapped to N different scalar values by taking inner products with a random vector (e.g., standard Gaussian). Hence, memorizing finite input-label pairs in d_x-dimension can be easily translated into memorizing finite input-label pairs in *one*-dimension. In other words, the input dimensionality d_x is not very important as long as they can be translated to distinct scalar values. In contrast, there is no "natural" way to design an injection from d_x-dimensional unit cube to lower dimension. Namely, to not lose the "information" from inputs, width d_x is required. Therefore, an arbitrary function cannot be approximated by the network of width independent of d_x.

4.2. Sufficient conditions for identifying memorization power

While Theorem 1 constructs some *tailor-made* network architectures for memorization, the following theorem states sufficient conditions for verifying the memorization power of *any given* network architectures. The proof of Theorem 3 is presented in Appendix E.

Theorem 3 *For any sigmoidal activation function σ, let f_θ be a σ network of L hidden layers having $d_\ell \geq 3$ neurons at the ℓ-th hidden layer. Then, for any $C, N, d_x \in \mathbb{N}$ and $\Delta > 1$, f_θ can memorize any Δ-separated set of N pairs in d_x dimension with C classes if the following statement holds:*

There exist $0 < L_1 < \cdots < L_K < L$ for some $2 \leq K \leq \log N$ satisfying the conditions below.

1. $\prod_{\ell=1}^{L_1} \lfloor (d_\ell + 1)/2 \rfloor \geq \Delta \sqrt{2\pi d_x}$.

2. $\sum_{\ell=L_{i-1}+1}^{L_i} (d_\ell - 2) \geq 2^{i+3}$ for all $1 < i \leq K - 1$.

3. $2^K \cdot \left(\sum_{\ell=L_{K-1}+1}^{L_K} (d_\ell - 2) \right) \cdot \left\lfloor \frac{L-L_K}{2\lceil \log C \rceil + 1} \right\rfloor \geq N^2 + 4.$

Our conditions in Theorem 3 require that the layers of the network can be "partitioned" into $K + 1$ distinct parts characterized by $L_1, \ldots, L_K$ for some $K \geq 2$. The partition into $K + 1$ parts corresponds to conditions in Theorem 3: The first part is for the first condition, the second to the $(K-1)$-th parts are for the second condition, and the remaining parts are for the third. We postpone the discussion on the role of each condition to Section 5. In Theorem 3, one can observe that the choice of K affects the second condition and the third condition. In particular, larger K implies that more hidden neurons are required to satisfy the second condition but less hidden neurons are required to satisfy the third condition. We discuss how K affects these two conditions in more detail later in this section.

Now we discuss the three conditions in Theorem 3 in detail. The first condition considers the first L_1 hidden layers. In order to satisfy this condition, deep and narrow architectures are better than shallow and wide architectures under a similar number of parameters due to the product form $\prod_{\ell=1}^{L_1} \lfloor (d_\ell + 1)/2 \rfloor$. Nevertheless, the architecture of the first L_1 hidden layers is not very critical since only $\log(\Delta \sqrt{2\pi d_x})$ layers are sufficient even with width 3 (e.g., $\log \Delta < 17$ for the ImageNet dataset).

The second condition requires that the number of hidden neurons $\sum_{\ell=L_{i-1}+1}^{L_i} (d_\ell - 2)$ in each *part* of layers is larger than or equal to some threshold value 2^{i+3}. In particular, the second condition is closely related to the third condition due to the following trade-off: As K increases, the LHS in the third condition increases, i.e., increasing K by one decreases the required neurons/layers after the L_{K-1}-th layer for satisfying the third condition by half. However, the second condition requires more hidden neurons as K grows. Simply put, increasing K by one requires doubling hidden neurons from the $(L_1 + 1)$-th hidden layer to the L_{K-1}-th hidden layer. Nevertheless, this doubling hidden neurons makes the LHS of the third condition double as well. In other words, due to the trade-off between the second condition and the third condition, there will be some optimal choice of K (and $L_1, \ldots, L_K$) maximizing the number of memorizable pairs for a given network.

The third condition is simple. As we explained, 2^K is approximately proportional to the number of hidden neurons from the $(L_1 + 1)$-th hidden layer to the L_{K-1}-th hidden layer. The second term in the LHS of the third condition $\sum_{\ell=L_{K-1}+1}^{L_K} (d_\ell - 2)$ simply counts the hidden neurons in the K-th part. On the other hand, the last term counts the number of layers in the $(K + 1)$-th part. This indicates that to satisfy conditions in Theorem 3 using few parameters, the last layers of the network should be deep and narrow. In particular, we note that such a deep and narrow architecture in the last layers is indeed necessary for RELU networks to memorize with $o(N/\log N)$ parameters (Bartlett et al., 2019), as we discussed in Section 1.1.

Deriving Theorem 2 using Theorem 3. Now, we describe how to show memorization with a sublinear number of parameters using our sufficient conditions by deriving Theorem 2 from Theorem 3. Consider a network of width 3, i.e., $d_\ell = 3$ for all ℓ. As we explained, the first conditions can be easily satisfied using $\Theta(\log d_x + \log \Delta)$ layers. For the second condition, consider choosing $K = \log(N^{2/3})$, then $\Theta(N^{2/3})$ hidden neurons (i.e., $\Theta(N^{2/3})$ hidden layers) would be sufficient, i.e., $L_{K-1} - L_1 = \Theta(N^{2/3})$. Finally, we choose L_K and L to satisfy $L_K - L_{K-1} = \Theta(N^{2/3})$ and $L - L_K = \Theta(N^{2/3} \log C)$. Then, it naturally satisfies the third condition using only $\Theta(N^{2/3} \log C)$ layers and completes the derivation of Theorem 2.

4.3. Discussions on Theorems 1–3

Extension to regression problem. The results of Theorems 1–3 can be easily applied to the regression problem, i.e., when labels are from $[0, 1]$. This is because one can simply translate the regression problem with some $\varepsilon > 0$ error tolerance to the classification problem with $\lceil 1/\varepsilon \rceil$ classes. Here, each class $c \in \{0, 1, \ldots, \lceil 1/\varepsilon \rceil - 1\}$ corresponds to the target value $c \cdot \varepsilon$. Hence, the regression problem can also be solved within ε error with $o(N)$ parameters, where the sufficient number of layers and the sufficient number of parameters are identical to the numbers in Theorems 1–3 with the replacement of $\log C$ with $\log(1/\varepsilon)$.

Relation with benefits of depth in neural networks. Our observation that deeper RELU networks have more memorization power is closely related to the recent studies on the benefits of depth in neural networks (see Section 2). While our observation indicates that depth is critical for the memorization power, these works mostly focused on showing the importance of depth for approximating functions. Here, the existing results on the benefits of depth for function approximation cannot directly imply the benefits of depth for memorization since they often focus on specific classes of functions or require parameters far more than $O(N)$.

On parameter precision. As in many works on the expressive power of neural networks (Cybenko, 1989; Pinkus, 1999; Lu et al., 2017; Bartlett et al., 2019; Yun et al., 2019; Park et al., 2021), we also assume real-valued parameters (i.e., weights and biases) rather than parameters of constantly bounded precision (e.g., binary, 16 bit floating-point numbers). Notably, parameters of constantly bounded precision is provably *insufficient* for memorization with $o(N)$ parameters: Any network of $o(N)$ parameters of constantly bounded precision cannot memorize any set of N inputs with arbitrary labels (Shalev-Shwartz and Ben-David, 2014). Hence, parameters of $\omega(1)$ precision (e.g., real-valued parameters) are *necessary* for memorization with $o(N)$ parameters. We note that the maximum required precision for Theorem 1 is $O(N^{2/3})$ at $w = 2/3$, i.e., memorization with $O(N^{2/3})$ parameters. This required precision decreases to $\Theta(1)$ as w goes to 1.

5. Network construction for proof of Theorem 1

In this section, we give a constructive proof of Theorem 1. First, we sketch the main proof idea in Section 5.1. Then, we provide the formal proof of Theorem 1 in Section 5.2. Lastly, we describe connections between our network construction and conditions in Theorem 3 in Section 5.3.

5.1. Proof outline

Before introducing the formal proof, we first sketch our network construction. In our proof, we mainly focus on constructing a STEP + ID network[6] which can memorize any Δ-separated N pairs in d_x dimension with C classes using $O\left(\log d_x + \log \Delta + \frac{N^{2-2w}}{1+(1.5w-1)\log N} \log C\right)$ hidden layers and $O\left(d_x + \log \Delta + N^w + N^{1-w/2} \log C\right)$ parameters, instead of designing a target σ network directly. Then, we approximate this STEP + ID network using a σ network of the same architecture to complete the proof.

    We first briefly discuss the motivation of our construction which can memorize N pairs with $o(N)$ parameters. Our construction is motivated by the RELU network construction achieving

---

6. Recall from Section 3 that STEP : $x \mapsto \mathbf{1}[x \geq 0]$ and ID : $x \mapsto x$.
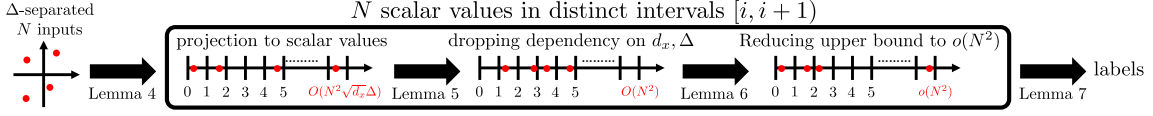
Figure 1: Illustration of the network construction for the proof of Theorem 1.

nearly tight VC-dimension (Bartlett et al., 2019): We observe that there exists a network of $o(N)$ parameters that can memorize any *well-separated* N scalar values in $[0, o(N^2))$ with arbitrary labels (Lemma 7 in Section 5.2). Here, the *well-separateness* means that each scalar value falls into each interval $[i, i+1)$. Namely, once we map input vectors to well-separated values in $[0, o(N^2))$ using $o(N)$ parameters, then we can memorize these input vectors with arbitrary labels with $o(N)$ parameters.

Under our motivation, we now describe a high-level outline of our STEP + ID network construction. First, we project Δ-separated, d_x-dimensional input vectors to well-separated scalar values in some bounded interval $[0, O(N^2\sqrt{d_x}\Delta))$, using $O(1)$ hidden layers and $O(d_x)$ parameters. In the following layers, we gradually decrease the upper bound on these scalar values from $O(N^2\sqrt{d_x}\Delta)$ to $O(N^{2-w})$ (i.e., $o(N^2)$) by relocating these values while preserving their well-separateness. This step utilizes $O(\log \Delta + \log d_x + \log N)$ hidden layers and $O(\log \Delta + \log d_x + N^w)$ parameters. Here, note that under the well-separateness, there exists one-to-one correspondence between these scalar values and input vectors. Finally, we map these scalar values in $[0, O(N^{2-w}))$ to corresponding labels using $O\left(\frac{N^{2-2w}}{1+(1.5w-1)\log N}\log C\right)$ hidden layers and $O(N^w + N^{1-w/2}\log C)$ parameters. By approximating this STEP + ID network construction using a σ network, we obtain the statement of Theorem 1. The overall network construction is illustrated in Figure 1.

5.2. Constructing target network

Now, we formally describe our network construction. To this end, we first construct the network using STEP and ID activation functions and approximate this network using a σ network.

Projecting input vectors to scalar values. We first map input vectors to bounded well-separated scalar values using the following lemma. The proof of Lemma 4 is presented in Appendix D.3.

Lemma 4 *For any Δ-separated $\mathcal{X} \in \binom{\mathbb{R}^{d_x}}{N}$, there exist $v \in \mathbb{R}^{d_x}$, $b \in \mathbb{R}$ such that*

$$\left\lfloor \{v^\top x + b : x \in \mathcal{X}\} \right\rfloor \in \binom{[O(N^2\sqrt{d_x}\Delta)]}{N}.$$

Lemma 4 states that any Δ-separated N vectors can be mapped to some well-separated scalar values bounded by $O(N^2\sqrt{d_x}\Delta)$ using a simple projection with $O(1)$ hidden layers and $O(d_x)$ parameters.

Decreasing upper bound on scalar values to $O(N^{2-w})$. We map these well-separated N values upper bounded by $O(N^2\sqrt{d_x}\Delta)$ to another well-separated values of the smaller upper bound $\lfloor N^2/4 + 1 \rfloor$ using the following lemma. The proof of Lemma 5 is presented in Appendix D.4.

Lemma 5 *For any $\mathcal{X} \in \binom{\mathbb{R}}{N}$ and $K \in \mathbb{N}$ such that $\lfloor \mathcal{X} \rfloor \in \binom{[K]}{N}$, there exists a STEP + ID network f of 1 hidden layer and width 3 such that $\lfloor f(\mathcal{X}) \rfloor \in \binom{[T]}{N}$ where $T := \max\left\{ \lceil K/2 \rceil, \lfloor N^2/4 + 1 \rfloor \right\}$.*

Lemma 5 states that $O(1)$ hidden layers and $O(1)$ parameters are sufficient for halving the upper bound until $\lfloor N^2/4 + 1 \rfloor$. Therefore, by applying Lemma 5 $O(\log \Delta + \log d_x)$ times, a network of $O(\log \Delta + \log d_x)$ hidden layers and $O(\log \Delta + \log d_x)$ parameters can decrease the upper bound $O(N^2 \sqrt{d_x} \Delta)$ to $\lfloor N^2/4 + 1 \rfloor$. The intuition behind Lemma 5 is that if the number of target intervals T is large enough compared to the number of inputs N, then inputs can be easily mapped without harming the well-separateness (i.e., no two inputs are mapped into the same interval $[i, i+1)$) using some simple network of a small number of layers and parameters. In particular, we construct the simple network f in the proof of Lemma 5 as

$$f(x) := \begin{cases} x & \text{if } x \in [0, T) \\ (x + b) \bmod T & \text{if } x \in [T, K) \end{cases} \tag{1}$$

for some $b \in [T]$. However, if T is not large enough compared to N (i.e., $T < \lfloor N^2/4 + 1 \rfloor$), then outputs of our network (1) may not preserve the well-separateness, i.e., $b \in [T]$ satisfying $f(\lfloor \mathcal{X} \rfloor) \in \binom{[T]}{N}$ may not exist. However, the upper bound on the well-separated values can be further decreased by utilizing more parameters, as stated in the following lemma. The proof of Lemma 6 is presented in Section D.5.

Lemma 6 *For any $\mathcal{X} \in \binom{\mathbb{R}}{N}$ and $K \in \mathbb{N}$ such that $\lfloor \mathcal{X} \rfloor \in \binom{[K]}{N}$, there exists a STEP + ID network f of 1 hidden layer and width $O(N^2/K)$ such that $\lfloor f(\mathcal{X}) \rfloor \in \binom{[T]}{N}$ where $T := \max \{\lceil K/2 \rceil, N\}$.*

The network in Lemma 6 can halve the upper bound K of the inputs beyond $\lfloor N^2/4 + 1 \rfloor$. However, the required number of parameters will be doubled if the current upper bound K decreases by half. Hence, in order to decrease the upper bound from $\lfloor N^2/4 + 1 \rfloor$ to $O(N^{2-w})$ using $O(\log N)$ applications of Lemma 6, we need $O(\log N)$ hidden layers and $O(N^w)$ parameters. Here, we construct each application of Lemma 6 using two hidden layers. One hidden layer of $O(N^2/K)$ hidden neurons implements f in Lemma 6. The other hidden layer of one hidden neuron just bypasses the one-dimensional output of f implemented in the previous layer. The reason behind using an extra layer is that, if we implement each application of Lemma 6 using a single hidden layer (as we did for Lemma 5), then it will introduce $O(N^4/K^2)$ parameters between each pair of adjacent hidden layers of width $O(N^2/K)$; we avoid these unnecessary parameters by adding extra layers of width 1.

Mapping scalar values to corresponding labels. So far, we have N well-separated values, upper bounded by $O(N^{2-w})$. Now, we map these values to corresponding labels (i.e., labels of their original inputs) using the following lemma, motivated by the RELU network achieving nearly tight VC-dimension (Bartlett et al., 2019). The proof of Lemma 7 is presented in Appendix D.6.

Lemma 7 *For any $C, V \in \mathbb{N}$ and $p \in [1/2, 1]$, there exists a STEP + ID network f_θ of $O(V^p + V^{1/2} \log C)$ parameters and $O\left(\frac{V^{1-p}}{1+(p-0.5)\log V}\right)$ hidden layers such that f_θ can memorize any $\mathcal{X} \in \binom{\mathbb{R}}{V}$ satisfying $\lfloor \mathcal{X} \rfloor = [V]$ with arbitrary labels in $[C]$.*

If we set $w \in [2/3, 1]$, $V = O(N^{2-w})$, and $p = \frac{w}{2-w}$ in Lemma 7, then $O\left(\frac{N^{2-2w}}{1+(1.5w-1)\log N} \log C\right)$ hidden layers and $O(N^w + N^{1-w/2} \log C)$ parameters are sufficient for mapping any N well-separated values in $[0, O(N^{2-w}))$ to corresponding labels.

Overall, our STEP + ID network construction mapping Δ-separated, d_x-dimensional vectors to well-separated values bounded by $O(N^{2-w})$ requires $O\left(\log d_x + \log \Delta + \frac{N^{2-2w}}{1+(1.5w-1)\log N} \log C\right)$ hidden layers and $O\left(d_x + \log \Delta + N^w + N^{1-w/2} \log C\right)$ parameters by Lemmas 4–7.
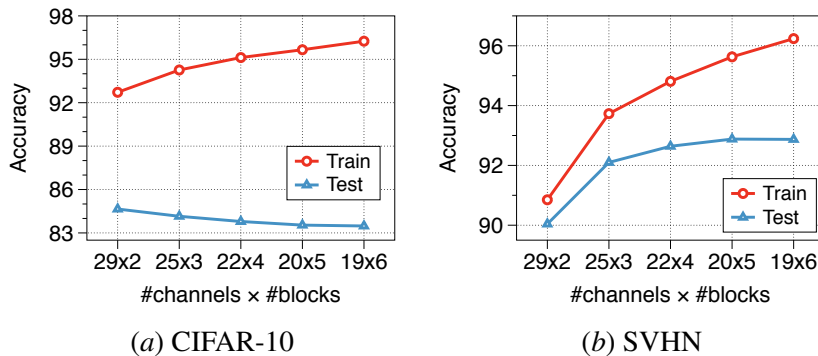
Figure 2: Depth-width trade-off under a similar number of parameters.

Approximating STEP + ID network using σ network. Finally, we approximate our STEP + ID network construction by a σ network of the same architecture, within any approximation error using the following lemma. The proof of Lemma 8 is presented in Appendix D.2.

Lemma 8 *For any finite set of inputs $\mathcal{X}$, $\varepsilon > 0$, STEP + ID network f, and sigmoidal activation function σ, there exists a σ network g having the same architecture with f satisfying $|f(x) - g(x)| < \varepsilon$ for all $x \in \mathcal{X}$.*

5.3. Functions of conditions in Theorem 3

The three conditions in Theorem 3 correspond to network constructions in Section 5.2. The first condition in Theorem 3 is for mapping Δ-separated inputs to some well-separated values bounded by $O(N^2)$. Here, decreasing the upper bound on the well-separated set by half does not require a large number of layers and parameters until the upper bound reaches $\lfloor N^2/4 + 1 \rfloor$ as stated in Lemma 5. The second condition in Theorem 3 is for improving $O(N^2)$ upper bound on the well-separated values to $O(N^2/2^K)$. As stated in Lemma 6, decreasing the upper bound on the well-separated set by half (i.e., increasing K by one in the second condition) requires twice as many parameters in the second condition. Finally, the third condition in Theorem 3 is for mapping well-separated values to their labels, which corresponds to Lemma 7. We refer Appendix E for more detailed descriptions.

6. Experiments

In this section, we study the effect of depth and width through experiments. In particular, we empirically verify whether our theoretical finding extends to practices: Do deep and narrow networks have more memorization power (i.e., better training accuracy) compared to their shallow and wide counterparts under similar numbers of parameters? For the experiments, we use residual networks (He et al., 2016) having the same number of channels for each layer. The detailed experimental setups are presented in Appendix A. In our experiments, we report the training and test accuracy of networks by varying the number of channels and the number of residual blocks.

6.1. Depth-width trade-off in memorization

We verify the memorization power of different network architectures having similar numbers of parameters. Figure 2 illustrates training and test accuracy of five different architectures with approxi-
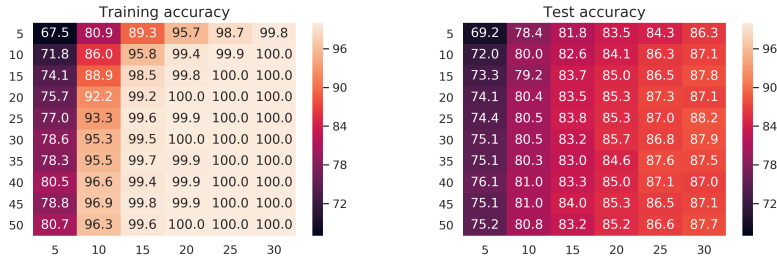
Training accuracy

	5	10	15	20	25	30
5	67.5	80.9	89.3	95.7	98.7	99.8
10	71.8	86.0	95.8	99.4	99.9	100.0
15	74.1	88.9	98.5	99.8	100.0	100.0
20	75.7	92.2	99.2	100.0	100.0	100.0
25	77.0	93.3	99.6	99.9	100.0	100.0
30	78.6	95.3	99.5	100.0	100.0	100.0
35	78.3	95.5	99.7	99.9	100.0	100.0
40	80.5	96.6	99.4	99.9	100.0	100.0
45	78.8	96.9	99.8	99.9	100.0	100.0
50	80.7	96.3	99.6	100.0	100.0	100.0

Test accuracy

	5	10	15	20	25	30
5	69.2	78.4	81.8	83.5	84.3	86.3
10	72.0	80.0	82.6	84.1	86.3	87.1
15	73.3	79.2	83.7	85.0	86.5	87.8
20	74.1	80.4	83.5	85.3	87.3	87.1
25	74.4	80.5	83.8	85.3	87.0	88.2
30	75.1	80.5	83.2	85.7	86.8	87.9
35	75.1	80.3	83.0	84.6	87.6	87.5
40	76.1	81.0	83.3	85.0	87.1	87.0
45	75.1	81.0	84.0	85.3	86.5	87.1
50	75.2	80.8	83.2	85.2	86.6	87.7

Figure 3: Training/test accuracy of networks for training the CIFAR-10 dataset by varying the number of channels (x-axis) and the number of residual blocks (y-axis).

mately 50000 parameters for classifying the CIFAR-10 dataset (Krizhevsky, 2009) and the SVHN dataset (Netzer et al., 2011). One can observe that as the network architecture becomes deeper and narrower, the training accuracy increases. Namely, deep and narrow networks memorize better than shallow and wide networks under similars number of parameters. This observation agrees with Theorem 1, which states that increasing depth reduces the required number of parameters for memorizing the same number of pairs.

However, more memorization power does not always imply better generalization (i.e., test accuracy). In Figure 2(b), as the depth increases, the test accuracy also increases for the SVHN dataset. In contrast, the test accuracy decreases for the CIFAR-10 dataset as the depth increases in Figure 2(a). In other words, overfitting occurs for the CIFAR-10 dataset while classifying the SVHN data receive benefits from more memorization power. Note that a similar observation has also been made in the recent double descent phenomenon (Belkin et al., 2019; Nakkiran et al., 2020) that more memorization power can both hurt/improve the generalization.

6.2. Effect of width and depth

In this section, we observe the effect of depth and width by varying both. Figure 3 reports the training and test accuracy for the CIFAR-10 dataset by varying the number for channels from 5 to 30 and the number of residual blocks from 5 to 50. We present the experimental results for the SVHN dataset in Appendix B under the same setup. First, we observe that networks of 15 channels with feature map size 32×32 successfully memorize (i.e., training accuracy over 99%). This size is much narrower than modern network architectures, e.g., ResNet-18 has 64 channels at the first hidden layer (He et al., 2016). On the other hand, too narrow networks (e.g., 5-channels) fail to memorize. This result does not contradict Theorem 2 as the test of memorization in experiments/practice involves the stochastic gradient descent. We note that similar observations are made for the SVHN dataset.

Furthermore, once the network memorize, we observe that increasing width is more effective than increasing depth for improving test accuracy. These results indicate that width is not very critical for the memorization power; however, it can be effective for generalization. Note that similar connections between generalization and the width/depth have also been made (Zhang et al., 2017; Golubeva et al., 2021).

7. Conclusion

In this paper, we prove that $\Theta(N^{2/3})$ parameters suffice for memorizing arbitrary N pairs under the mild Δ-separateness condition. Our result provides significantly improved results, compared to the prior results showing the sufficiency of $\Theta(N)$ parameters with/without conditions on pairs. Theorem 1 shows that deeper networks have more memorization power. This result coincides with the recent study on the benefits of depth for function approximation. On the other hand, Theorem 2 shows that network width is not important for the memorization power. We also provide sufficient conditions for identifying the memorization power of networks. Finally, we empirically confirm our theoretical results.

Acknowledgments

References

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*, 2018.

Peter L. Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 2019.

Eric B. Baum. On the capabilities of multilayer perceptrons. *Journal of Complexity*, 4(3):193–215, 1988. ISSN 0885-064X.

Eric B. Baum and David Haussler. What size net gives valid generalization? In *Advances in Neural Information Processing Systems*, 1989.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Vaggos Chatziafratis, Sai Ganesh Nagarajan, and Ioannis Panageas. Better depth-width trade-offs for neural networks through the lens of dynamical systems. In *International Conference on Machine Learning*, 2020a.

Vaggos Chatziafratis, Sai Ganesh Nagarajan, Ioannis Panageas, and Xiao Wang. Depth-width trade-offs for relu networks via sharkovsky's theorem. In *International Conference on Learning Representations*, 2020b.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on Learning Theory*, 2016.

Walter Gautschi. Some elementary inequalities relating to the gamma and incomplete gamma function. *Journal of Mathematics and Physics*, 38(1-4):77–81, 1959.

Anna Golubeva, Guy Gur-Ari, and Behnam Neyshabur. Are wider nets better given the same number of parameters? In *International Conference on Learning Representations*, 2021.

Boris Hanin and Mark Sellke. Approximating continuous functions by ReLU nets of minimal width. *arXiv preprint arXiv:1710.11278*, 2017.

Moritz Hardt and Tengyu Ma. Identity matters in deep learning. In *International Conference on Learning Representations*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Guang-Bin Huang. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks*, 14(2):274–281, 2003.

Guang-Bin Huang and Haroon A Babri. Upper bounds on the number of hidden neurons in feed-forward networks with arbitrary bounded nonlinear activation functions. *IEEE Transactions on Neural Networks*, 9(1):224–229, 1998.

Jesse Johnson. Deep, skinny neural networks are not universal approximators. In *International Conference on Learning Representations*, 2019.

Marek Karpinski and Angus Macintyre. Polynomial bounds for VC dimension of sigmoidal and general Pfaffian neural networks. *Journal of Computer and System Sciences*, 54(1):169–176, 1997.

Patrick Kidger and Terry Lyons. Universal approximation with deep narrow networks. In *Conference on Learning Theory*, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems*, 2017.

Wolfgang Maass. Bounds for the computational power and learning complexity of analog neural nets. *SIAM Journal on Computing*, 26(3):708–732, 1997.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020.

Yuval Netzer, T. Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Deep Learning and Unsupervised Feature Learning Workshop*, 2011.

Quynh Nguyen and Matthias Hein. Optimization landscape and expressivity of deep CNNs. In *International Conference on Machine Learning*, 2018.

Sejun Park, Chulhee Yun, Jaeho Lee, and Jinwoo Shin. Minimum width for universal approximation. In *International Conference on Learning Representations*, 2021.

Allan Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8: 143–195, 1999.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Eduardo D. Sontag. Shattering all sets of k points in "general position" requires (k—1)/2 parameters. *Neural Computation*, 9(2):337–348, 1997.

Matus Telgarsky. Benefits of depth in neural networks. In *Conference on Learning Theory*, 2016.

Roman Vershynin. Memory capacity of neural networks with threshold and ReLU activations. *arXiv preprint 2001.06938*, 2020.

Masami Yamasaki. The lower bound of the capacity for a neural network with multiple hidden layers. In *International Conference on Artificial Neural Networks*, 1993.

Dmitry Yarotsky. Optimal approximation of continuous functions by very deep relu networks. In *Conference on Learning Theory*, 2018.

Chulhee Yun, Suvrit Sra, and Ali Jadbabaie. Small ReLU networks are powerful memorizers: a tight analysis of memorization capacity. In *Advances in Neural Information Processing Systems*, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

Appendix A. Experimental setup

In this section, we described the details on residual network architectures and hyper-parameter setups used for our experiments.

We use the residual networks of the following structure. First, a convolutional layer and RELU maps a 3-channel input image to a c-channel feature map. Here, the size of the feature map is identical to the size of input images. Then, we apply b residual blocks where each residual block maps $x \mapsto \text{RELU}(\text{CONV} \circ \text{RELU} \circ \text{CONV}(x) + x)$ while preserving the number of channels and the size of feature map. Finally, we apply an average pooling layer and a fully-connected layer. We train the model for 5×10^5 iterations with batch size 64 by the stochastic gradient descent. We use the initial learning rate 0.1, weight decay 10^{-4}, and the learning rate decay at the 1.5×10^5-th iteration and the 3.5×10^5-th iteration by a multiplicative factor 0.1. All presented results are averaged over three independent trials.

Appendix B. Training and test accuracy for SVHN dataset
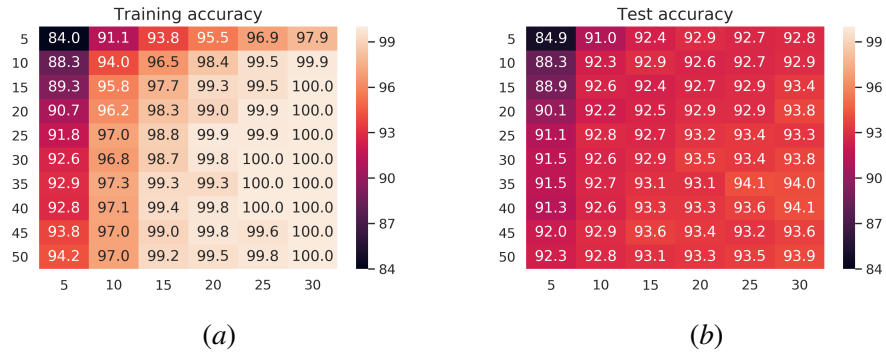

(a)

(b)

Figure 4: Training/test accuracy of networks for training the SVHN dataset by varying the number of channels (x-axis) and the number of residual blocks (y-axis).

Appendix C. Δ-separateness of Gaussian random vectors

While we mentioned in Section 1.1 that digital nature of data enables the Δ-separateness of inputs with small Δ, random inputs are also Δ-separated with small Δ with high probability. In particular, we prove the following lemma.

Lemma 9 *For any $d_x \in \mathbb{N}$, consider a set of N vectors $\mathcal{X} = \{x_1, \ldots, x_N\} \subset \mathbb{R}^{d_x}$ where each entry of x_i is drawn from the i.i.d. standard normal distribution. Then, for any $\delta > 0$, $\mathcal{X}$ is $\left((N/\sqrt{\delta})^{2/d_x}\sqrt{3e + \frac{5e}{d_x}\ln(N/\sqrt{\delta})}\right)$-separated with probability at least $1 - \delta$.*

Lemma 9 implies that Theorem 1 and Theorem 2 can be successfully applied to random Gaussian input vectors as the separateness condition in Lemma 9 is much weaker than our $2^{O(N^{2/3})}$-separateness condition for memorization with $o(N)$ parameters.

Proof First, notice that for $i \neq j$, we have

$$\mathbb{P}\left(\sqrt{\frac{2}{e}} \cdot \left(\frac{\sqrt{\delta}}{N}\right)^{2/d_x} \leq \frac{1}{\sqrt{d_x}}\|x_i - x_j\|_2 \leq \sqrt{6 + \frac{10}{d_x}\ln\frac{N}{\sqrt{\delta}}}\right)$$

$$= \mathbb{P}\left(\frac{2}{e} \cdot \left(\frac{\sqrt{\delta}}{N}\right)^{4/d_x} \leq \frac{1}{d_x}\|x_i - x_j\|_2^2 \leq 6 + \frac{10}{d_x}\ln\frac{N}{\sqrt{\delta}}\right)$$

$$= \mathbb{P}\left(\frac{2}{e} \cdot \left(\frac{\sqrt{\delta}}{N}\right)^{4/d_x} \leq \frac{2}{d_x}X \leq 6 + \frac{10}{d_x}\ln\frac{N}{\sqrt{\delta}}\right)$$

$$= 1 - \mathbb{P}\left(\frac{2}{e} \cdot \left(\frac{\sqrt{\delta}}{N}\right)^{4/d_x} > \frac{2}{d_x}X\right) - \mathbb{P}\left(\frac{2}{d_x}X > 6 + \frac{10}{d_x}\ln\frac{N}{\sqrt{\delta}}\right)$$

$$= 1 - \mathbb{P}\left(\frac{1}{e} \cdot \left(\frac{\sqrt{\delta}}{N}\right)^{4/d_x} \cdot d_x > X\right) - \mathbb{P}\left(X > \left(3 + \frac{5}{d_x}\ln\frac{N}{\sqrt{\delta}}\right) \cdot d_x\right)$$

$$\geq 1 - \left(\frac{1}{e} \cdot \left(\frac{\sqrt{\delta}}{N}\right)^{4/d_x} e^{1 - \frac{1}{e} \cdot (\frac{\sqrt{\delta}}{N})^{4/d_x}}\right)^{d_x/2} - \left(\left(3 + \frac{5}{d_x}\ln\frac{N}{\sqrt{\delta}}\right)e^{-2}\left(\frac{\sqrt{\delta}}{N}\right)^{5/d_x}\right)^{d_x/2}$$

$$= 1 - \frac{\delta}{N^2} - \frac{\delta}{N^2} \cdot \left(\left(3 + \frac{5}{d_x}\ln\frac{N}{\sqrt{\delta}}\right)e^{-2}\left(\frac{\sqrt{\delta}}{N}\right)^{1/d_x}\right)^{d_x/2}$$

$$\geq 1 - \frac{\delta}{N^2} - \frac{\delta}{N^2} \cdot \left(\frac{3}{e^2 N^{1/d_x}} + \frac{5}{e^2} \cdot \frac{\ln\left(\frac{N}{\sqrt{\delta}}\right)^{1/d_x}}{\left(\frac{N}{\sqrt{\delta}}\right)^{1/d_x}}\right)^{d_x/2}$$

$$\geq 1 - \frac{\delta}{N^2} - \frac{\delta}{N^2} \cdot \left(\frac{3}{e^2} + \frac{5}{e^3}\right)^{d_x/2}$$

$$\geq 1 - \frac{\delta}{N^2} - \frac{\delta}{N^2} = 1 - \frac{2\delta}{N^2} \tag{2}$$

where X denotes a chi-square random variable with d_x degrees of freedom. For the first inequality in (2), we use the inequalities

$$\mathbb{P}(X < z \cdot d_x) \leq \inf_{t>0} \frac{\mathbb{E}[e^{-tX}]}{e^{-t(z \cdot d_x)}} = \inf_{t>0} \frac{(1+2t)^{-d_x/2}}{e^{-t(z \cdot d_x)}} = (ze^{1-z})^{d_x/2} \text{ for } 0 < z < 1$$

$$\mathbb{P}(X > z \cdot d_x) \leq \inf_{t>0} \frac{\mathbb{E}[e^{tX}]}{e^{t(z \cdot d_x)}} = \inf_{0<t<1/2} \frac{(1-2t)^{-d_x/2}}{e^{t(z \cdot d_x)}} = (ze^{1-z})^{d_x/2} \text{ for } z > 1$$

which directly follow from the Chernoff bound for the chi-square distribution. For the third inequality in (2), we use the fact that $\max_{x>0}(\ln x)/x = 1/e$ and $N \geq 1$. For the last inequality in (2), we use $\frac{3}{e^2} + \frac{5}{e^3} < 1$.

Then, $\mathcal{X}$ is $\left((N/\sqrt{\delta})^{2/d_x} \sqrt{3e + \frac{5e}{d_x} \ln(N/\sqrt{\delta})}\right)$-separated with probability at least $1 - \delta$ since the following bound holds:

$$\mathbb{P}\left(\sqrt{\frac{2}{e}} \delta^{1/d_x} N^{-2/d_x} \leq \frac{1}{\sqrt{d_x}} \|x_i - x_j\|_2 \leq \sqrt{6 + \frac{10}{d_x} \ln \frac{N}{\sqrt{\delta}}}, \ \forall i \neq j\right)$$

$$= 1 - \mathbb{P}\left(\exists i \neq j \text{ s.t. } \|x_i - x_j\|_2 > \sqrt{6 + \frac{10}{d_x} \ln \frac{N}{\sqrt{\delta}}} \text{ or } \|x_i - x_j\|_2 < \sqrt{\frac{2}{e}} \delta^{1/d_x} N^{-2/d_x}\right)$$

$$\geq 1 - \sum_{i \neq j} \mathbb{P}\left(\|x_i - x_j\|_2 > \sqrt{6 + \frac{10}{d_x} \ln \frac{N}{\sqrt{\delta}}} \text{ or } \|x_i - x_j\|_2 < \sqrt{\frac{2}{e}} \delta^{1/d_x} N^{-2/d_x}\right)$$

$$= 1 - \sum_{i \neq j} \left(1 - \mathbb{P}\left(\sqrt{\frac{2}{e}} \delta^{1/d_x} N^{-2/d_x} \leq \frac{1}{\sqrt{d_x}} \|x_i - x_j\|_2 \leq \sqrt{6 + \frac{10}{d_x} \ln \frac{N}{\sqrt{\delta}}}\right)\right)$$

$$\geq 1 - \frac{N(N-1)}{2} \times \frac{2\delta}{N^2}$$

$$\geq 1 - \delta$$

where the first inequality follows from the union bound and the second inequality follows from (2). This completes the proof of Lemma 9. $\blacksquare$

Appendix D. Proof of Lemmas for Theorem 1

D.1. Tools for proving Lemma 8

We present the following claims for the proving lemmas. In particular, Claim 10 and Claim 11 guide how to approximate ID and STEP by a single sigmoidal neuron, respectively.

Claim 10 (Kidger and Lyons (2020, Lemma 4.1)) *For any sigmoidal activation function σ, for any bounded interval $\mathcal{I} \subset \mathbb{R}$ for any $\varepsilon > 0$, there exist $a, b, c, d \in \mathbb{R}$ such that $|a \cdot \sigma(c \cdot x + d) + b - x| < \varepsilon$ for all $x \in \mathcal{I}$.*

Claim 11 *For any sigmoidal activation function σ, for any $\varepsilon, \delta > 0$, there exist $a, b, c \in \mathbb{R}$ such that $|a \cdot \sigma(c \cdot x) + b - \mathbf{1}[x \geq 0]| < \varepsilon$ for all $x \notin [-\delta, \delta]$.*

Proof We assume that $\alpha := \lim_{x \to -\infty} \sigma(x) < \lim_{x \to \infty} \sigma(x) =: \beta$ where the case that $\beta < \alpha$ can be proved in a similar manner. From the definition of α, β, there exists $k > 0$ such that $|\sigma(x) - \alpha| < (\beta - \alpha)\varepsilon$ if $x < -k$ and $|\sigma(x) - \beta| < (\beta - \alpha)\varepsilon$ if $x > k$. Then, choosing $a = \frac{1}{\beta - \alpha}$, $b = -\frac{\alpha}{\beta - \alpha}$, and $c = \frac{k}{\delta}$ completes the proof of Claim 11. ∎

Claim 12 *For any $a, x \in \mathbb{R}$ such that $a \neq 0$, for any $b \in \mathbb{N}$, it holds that $\left\lceil \frac{x}{a \cdot b} \right\rceil = \left\lceil \frac{\lceil x/a \rceil}{b} \right\rceil$.*

Proof It trivially holds that $\left\lceil \frac{x}{a \cdot b} \right\rceil \leq \left\lceil \frac{\lceil x/a \rceil}{b} \right\rceil$. Now, we show a contradiction if $\left\lceil \frac{x}{a \cdot b} \right\rceil < \left\lceil \frac{\lceil x/a \rceil}{b} \right\rceil$. Suppose that $\left\lceil \frac{x}{a \cdot b} \right\rceil < \left\lceil \frac{\lceil x/a \rceil}{b} \right\rceil$. Then, there exists an integer m such that

$$\frac{x}{a \cdot b} \leq m < \frac{\lceil x/a \rceil}{b} \qquad \text{and hence,} \qquad \frac{x}{a} \leq \left\lceil \frac{x}{a} \right\rceil \leq b \cdot m < \left\lceil \frac{x}{a} \right\rceil$$

which leads to a contradiction. This completes the proof of Claim 12. ∎

D.2. Proof of Lemma 8

Without loss of generality, we first assume that for any network h having STEP neurons, all inputs to STEP neurons (i.e., $\mathbf{1}[x \geq 0]$) is non-zero for all $x \in \mathcal{X}$ during the evaluation of $h(x)$. This assumption can be easily satisfied by adding some small bias to the inputs of STEP neurons so that the output of STEP neurons does not change for all $x \in \mathcal{X}$. Note that such a bias always exists since $|\mathcal{X}| < \infty$. Furthermore, introducing this assumption does not affect to the values of $h(x)$ for all $x \in \mathcal{X}$

Now, we describe our construction of g. Let $\delta > 0$ be the minimum absolute value among all inputs to STEP neurons during the evaluation of $f(x)$ for all $x \in \mathcal{X}$. Let L be the number of hidden layers in f. Starting from f, we iteratively substitute the STEP and ID hidden neurons into σ hidden neurons, from the last hidden layer to the first hidden layer. In particular, by using Claim 10 and Claim 11, we replace STEP and ID neurons in the ℓ-th hidden layer by σ neurons approximating ID and STEP.

First, let g_L be a network identical to f except for its L-th hidden layer. In particular, the L-th hidden layer of g_L consists of σ neurons approximating STEP and ID neurons in the L-th hidden layer of f. Here, we accurately approximate STEP and ID neurons by σ using Claim 10 and Claim

11 so that $|g_L(x) - f(x)| < \varepsilon/L$ for all $x \in \mathcal{X}$. Note that such approximation always exists due to the existence of $\delta > 0$. Now, let g_{L-1} be a network identical to f_L except for its $(L-1)$-th hidden layer consisting of σ neurons approximating STEP and ID neurons in the $(L-1)$-th hidden layer of g_L. Here, we also accurately approximate STEP and ID neurons by σ using Claim 10 and Claim 11 so that $|g_L(x) - g_{L-1}(x)| < \varepsilon/L$ for all $x \in \mathcal{X}$. If we repeat this procedure until replacing the first hidden layer, then $g := g_1$ would be the desired network satisfying that $|f(x) - g(x)| < \varepsilon$ for all $x \in \mathcal{X}$. This completes the proof of Lemma 8.

D.3. Proof of Lemma 4

We let $d_x \geq 2$, as the lemma trivially holds when $d_x = 1$. Now, consider a projection $x \mapsto u^\top x \in \mathbb{R}$ by some vector $u \in \mathbb{R}^{d_x}$. There exists some u such that

$$\frac{\max_{\{x,x'\}\in\binom{\mathcal{X}}{2}} |u^\top(x - x')|}{\min_{\{x,x'\}\in\binom{\mathcal{X}}{2}} |u^\top(x - x')|} < N^2 \Delta \sqrt{\frac{\pi d_x}{8}}. \tag{3}$$

holds, as we can see from the following lemma. The proof of Lemma 13 is presented in Appendix F.1.

Lemma 13 *For any $N, d_x \in \mathbb{N}$, for any $\mathcal{X} \in \binom{\mathbb{R}^{d_x}}{N}$, there exists a unit vector $u \in \mathbb{R}^{d_x}$ such that* $\frac{1}{N^2}\sqrt{\frac{8}{\pi d_x}}\|x - x'\|_2 \leq |u^\top(x - x')| \leq \|x - x'\|_2$ *for all $x, x' \in \mathcal{X}$.*

Finally, we construct the desired map by

$$x \mapsto \frac{u^\top x - \min\{u^\top x : x \in \mathcal{X}\}}{\min_{\{x,x'\}\in\binom{\mathcal{X}}{2}} |u^\top(x - x')|} =: v^\top x + b.$$

Then, the following equalities/inequalities holds:

$$\min_{\{x,x'\}\in\binom{\mathcal{X}}{2}} |v^\top(x - x')| = \frac{\min_{\{x,x'\}\in\binom{\mathcal{X}}{2}} |u^\top(x - x')|}{\min_{\{x,x'\}\in\binom{\mathcal{X}}{2}} |u^\top(x - x')|} = 1,$$

$$\max_{\{x,x'\}\in\binom{\mathcal{X}}{2}} |v^\top(x - x')| = \frac{\max_{\{x,x'\}\in\binom{\mathcal{X}}{2}} |u^\top(x - x')|}{\min_{\{x,x'\}\in\binom{\mathcal{X}}{2}} |u^\top(x - x')|} \leq N^2 \Delta \sqrt{\frac{\pi d_x}{8}},$$

$$\min_{x\in\mathcal{X}} v^\top x + b = 0,$$

$$\max_{x\in\mathcal{X}} v^\top x + b = \max_{\{x,x'\}\in\binom{\mathcal{X}}{2}} |v^\top(x - x')| \leq N^2 \Delta \sqrt{\frac{\pi d_x}{8}},$$

i.e., $\lfloor\{v^\top x + b : x \in \mathcal{X}\}\rfloor \in \binom{\lceil\lceil N^2\Delta\sqrt{\pi d_x/8}\rceil\rceil}{N}$. This completes the proof of Lemma 4.

D.4. Proof of Lemma 5

To prove Lemma 5, we introduce its generalization as follows. The proof of Lemma 14 is presented in Appendix F.2.

Lemma 14 *For any $N, K, d \in \mathbb{N}$ and for any $\mathcal{X}$ such that $\lfloor \mathcal{X} \rfloor \in \binom{[K]}{N}$, there exists a STEP + ID network f of 1 hidden layer and width d such that $\lfloor f(\mathcal{X}) \rfloor \in \binom{[T]}{N}$ where $T := \max \left\{ \lceil \frac{K}{\lfloor (d+1)/2 \rfloor} \rceil, \lfloor \frac{N^2}{4} + 1 \rfloor \right\}$.*

From Lemma 14, one can observe that a STEP + ID network of 1 hidden layer consisting of 3 hidden neuron which can map any $\mathcal{Z}$ such that $\lfloor \mathcal{Z} \rfloor \in \binom{[K]}{N}$ to $\mathcal{Z}'$ such that $\lfloor \mathcal{Z}' \rfloor \in \binom{[T]}{N}$ where $T := \max \left\{ \lceil \frac{K}{\lfloor (d+1)/2 \rfloor} \rceil, \lfloor \frac{N^2}{4} + 1 \rfloor \right\}$. Combining Lemma 14 and Lemma 8 completes the proof of Lemma 5.

D.5. Proof of Lemma 6

To prove Lemma 6, we introduce its generalization as follows. The proof of Lemma 15 is presented in Appendix F.3.

Lemma 15 *For any $N, K, L, d_1, \ldots, d_L \in \mathbb{N}$ such that $d_\ell \geq 3$ for all ℓ, for any $\mathcal{X}$ such that $\lfloor \mathcal{X} \rfloor \in \binom{[K]}{N}$, there exists a STEP + ID network f of L hidden layers having d_ℓ neurons at the ℓ-th hidden layer such that $\lfloor f(\mathcal{X}) \rfloor \in \binom{[T]}{N}$ where $T := \min \left\{ K, \max \left\{ N \times \lceil N/C \rceil, \lceil K/2 \rceil \right\} \right\}$ and $C := \lfloor \frac{1}{2} + \frac{1}{2} \sum_{\ell=1}^{L} (d_\ell - 2) \rfloor$.*

We choose $C = \lceil 4N^2/K \rceil$ in Lemma 15 so that

$$N \times \lceil N/C \rceil \leq N \times \lceil K/(4N) \rceil \leq \max\{N, K/2\}.$$

Then, a STEP + ID network f of one hidden layer having $\Theta(N^2/K)$ hidden neurons can map any $\mathcal{X}$ such that $\lfloor \mathcal{X} \rfloor \in \binom{[K]}{N}$ to $f(\mathcal{X})$ such that $\lfloor f(\mathcal{X}) \rfloor = \binom{[T]}{N}$ where $T := \max\{N, \lceil K/2 \rceil\}$. Combining this and Lemma 8 completes the proof of Lemma 6.

D.6. Proof of Lemma 7

To prove Lemma 7, we introduce the following lemma. We note that Lemma 16 follows the construction for proving VC-dimension lower bound of RELU networks (Bartlett et al., 2019). The proof of Lemma 16 is presented in Appendix F.4.

Lemma 16 *For any $A, B, D, K, R \in \mathbb{N}$ such that $AB \geq K$, there exists a STEP + ID network f_θ of $2\lceil \frac{BD}{R} \rceil + 2$ hidden layers and $4A + \left((2R+5)2^R + 2R^2 + 8R + 7 \right) \lceil \frac{BD}{R} \rceil - R2^R - R^2 + 3$ parameters satisfying the following property: For any finite set $\mathcal{X} \subset [0, K)$, for any $y : [K] \to [2^D]$, there exists θ such that $f_\theta(x) = y(\lfloor x \rfloor)$ for all $x \in \mathcal{X}$.*

By choosing $K \leftarrow V, A \leftarrow \Theta(V^p), B \leftarrow \Theta(V^{1-p}), R \leftarrow 1 + (p - 0.5) \log V$, and $D \leftarrow \lceil \log C \rceil$ in Lemma 16, there exists a STEP + ID network of $O\left(\frac{V^{1-p}}{1+(p-0.5)\log V} \log C \right)$ hidden layers and $O(V^p + V^{1/2} \log C)$ parameters which can memorize any $\mathcal{X} \subset \mathbb{R}$ of size V with C classes satisfying $\lfloor \mathcal{X} \rfloor = [V]$. Combining this with Lemma 8 completes the proof of Lemma 7.

Appendix E. Proof of Theorem 3

The proof of Theorem 3 have the same structure of the proof of Theorem 1. In particular, we construct a STEP + ID network and use Lemma 8 as in the proofs of Theorem 1 and Theorem 2.

For the network construction, we divide the function of f_θ into four disjoint parts, as in Section 5. The first part does not utilize a hidden layers but construct project input vectors into scalar values. The second part corresponds to the first L_1 hidden layers decreases the upper bound on scalar values to $O(N^2)$. The third part corresponds to the next $L_{K-1} - L_1$ hidden layers further decreases the upper bound to $o(N^2)$. The last part corresponds to the rest hidden layers construct a network mapping hidden features to their labels.

Now, we describe our construction in detail. To begin with, let us denote a Δ-separated set of inputs by $\mathcal{X}_1$. First, from Lemma 13, one can project $\mathcal{X}_1$ to $\mathcal{X}_2$ such that $\lfloor \mathcal{X}_2 \rfloor \in \binom{[\lceil N^2 \Delta \sqrt{\pi d_x/8} \rceil]}{N}$. Note that the projection step does not require to use hidden layers as it can be absorbed into the linear map before the next hidden layer. Then, the first L_1 hidden layers can map $\mathcal{X}_2$ to $\mathcal{X}_3$ such that $\lfloor \mathcal{X}_3 \rfloor \in \binom{[\lfloor N^2/4+1 \rfloor]}{N}$ from Lemma 14 and the first condition:

$$\prod_{\ell=1}^{L_1} \left\lfloor \frac{d_\ell + 1}{2} \right\rfloor \ge \Delta \sqrt{2\pi d_x} \Rightarrow \prod_{\ell=1}^{L_1} \left\lfloor \frac{d_\ell + 1}{2} \right\rfloor \ge \frac{N^2 \Delta \sqrt{\pi d_x/8}}{N^2/4}$$

$$\Rightarrow \frac{N^2}{4} \ge \frac{N^2 \Delta \sqrt{\pi d_x/8}}{\prod_{\ell=1}^{L_1} \lfloor (d_\ell + 1)/2 \rfloor}$$

$$\Rightarrow \left\lfloor \frac{N^2}{4} + 1 \right\rfloor \ge \left\lceil \frac{N^2 \Delta \sqrt{\pi d_x/8}}{\prod_{\ell=1}^{L_1} \lfloor (d_\ell + 1)/2 \rfloor} \right\rceil.$$

Note that we also utilize Claim 12 for the sequential application of Lemma 14. Consecutively, the next $L_{K-1} - L_1$ hidden layers can map $\mathcal{X}_3$ to $\mathcal{X}_4$ such that $\lfloor \mathcal{X}_4 \rfloor \in \binom{[\lceil U/2^{K-2} \rceil]}{N}$ from Lemma 15 and the second condition:

$$\sum_{\ell=L_{i-1}+1}^{L_i} (d_\ell - 2) \ge 2^{i+3} \Rightarrow \frac{1}{2} \sum_{\ell=L_{i-1}+1}^{L_i} (d_\ell - 2) \ge 2^{i+2} \Rightarrow \left\lfloor \frac{1}{2} + \frac{1}{2} \sum_{\ell=L_{i-1}+1}^{L_i} (d_\ell - 2) \right\rfloor \ge 2^{i+2}$$

$$\Rightarrow \frac{N}{\left\lfloor \frac{1}{2} + \frac{1}{2} \sum_{\ell=L_{i-1}+1}^{L_i} (d_\ell - 2) \right\rfloor} \le \frac{N}{2^{i+2}}$$

$$\Rightarrow \left\lceil \frac{N}{\left\lfloor \frac{1}{2} + \frac{1}{2} \sum_{\ell=L_{i-1}+1}^{L_i} (d_\ell - 2) \right\rfloor} \right\rceil \le \frac{N}{2^{i+1}}$$

$$\Rightarrow N \times \left\lceil \frac{N}{\left\lfloor \frac{1}{2} + \frac{1}{2} \sum_{\ell=L_{i-1}+1}^{L_i} (d_\ell - 2) \right\rfloor} \right\rceil \le \frac{N^2/4}{2^{i-1}}$$

$$\Rightarrow N \times \left\lceil \frac{N}{\left\lfloor \frac{1}{2} + \frac{1}{2} \sum_{\ell=L_{i-1}+1}^{L_i} (d_\ell - 2) \right\rfloor} \right\rceil \le \left\lceil \frac{\lfloor N^2/4 + 1 \rfloor}{2^{i-1}} \right\rceil$$

where we use the inequality $\lceil a \rceil \le 2a$ for $a \ge 1/2$ and the assumption $K \le \log N$, i.e., $N/2^{i+1} \ge 1$ for $i \le \log N - 1$. Here, we also utilize Claim 12 for the sequential application of Lemma 15.

Finally, we reinterpret the Third condition as follows.

$$2^K \cdot \left(\sum_{\ell=L_{K-1}+1}^{L_K} (d_\ell - 2) \right) \cdot \left\lfloor \frac{L - L_K}{2\lceil \log C \rceil + 1} \right\rfloor \geq N^2 + 4$$

$$\Rightarrow \left(\sum_{\ell=L_{K-1}+1}^{L_K} (d_\ell - 2) \right) \cdot \left\lfloor \frac{L - L_K}{2\lceil \log C \rceil + 1} \right\rfloor \geq \frac{N^2/4 + 1}{2^{K-2}}$$

$$\Rightarrow \left(\sum_{\ell=L_{K-1}+1}^{L_K} (d_\ell - 2) \right) \cdot \left\lfloor \frac{L - L_K}{2\lceil \log C \rceil + 1} \right\rfloor \geq \left\lceil \frac{\lfloor N^2/4 + 1 \rfloor}{2^{K-2}} \right\rceil$$

Finally, from the following lemma and the above inequality, the rest hidden layers can map $\mathcal{X}_4$ to their corresponding labels (by choosing $L' := L_3$). The proof of Lemma 17 is presented in Appendix F.7.

Lemma 17 *For $D, K, L, d_1, \ldots, d_L \in \mathbb{N}$ such that $d_\ell \geq 3$ for all ℓ, suppose that there exist $0 < L' < L$ satisfying that*

$$\left(\sum_{\ell=1}^{L'} (d_\ell - 2) \right) \cdot \left\lfloor \frac{L - L'}{2D + 1} \right\rfloor \geq K.$$

Then, there exists a STEP + ID *network f_θ of L hidden layers having d_ℓ hidden neurons at the ℓ-th hidden layer such that for any finite $\mathcal{X} \subset [0, K)$, for any $y : [K] \to [2^D]$, there exists θ satisfying $f_\theta(x) = y(\lfloor x \rfloor)$ for all $x \in \mathcal{X}$.*

Appendix F. Proofs of technical lemmas

F.1. Proof of Lemma 13

We focus on proving the lower bound; the upper bound holds as $\|u\|_2 = 1$. We also let $N \geq 2$ and $d_x \geq 2$, as the lemma trivially holds for a singleton $\mathcal{X}$ or $d_x = 1$. We prove the lemma by showing that for any vector $v \in \mathbb{R}^{d_x}$, a *random* unit vector $u \in \mathbb{R}^{d_x}$ uniformly randomly drawn from the hypersphere $\mathbb{S}^{d_x-1}$ satisfies

$$\mathbb{P}\left(|u^\top v| < \frac{\|v\|_2}{N^2}\sqrt{\frac{8}{\pi d_x}}\right) < \frac{2}{N^2}. \tag{4}$$

Once we show (4), the *existence* of a unit vector u satisfying the lower bound of Lemma 13 follows. To see this, define $\mathcal{V} := \{x - x' \;:\; \{x, x'\} \in \binom{\mathcal{X}}{2}, x \leq x'\}$ for some total order $\leq$ on $\mathcal{X}$. Then, the union bound implies

$$\mathbb{P}\left(\bigcup_{v \in \mathcal{V}}\left\{|u^\top v| < \frac{\|v\|_2}{N^2}\sqrt{\frac{8}{\pi d_x}}\right\}\right) \leq \sum_{v \in \mathcal{V}} \mathbb{P}\left(|u^\top v| < \frac{\|v\|_2}{N^2}\sqrt{\frac{8}{\pi d_x}}\right) < \frac{N(N-1)}{2} \times \frac{2}{N^2} < 1,$$

and thus there exists at least one unit vector u that satisfies the lower bound.

To show (4), we begin by noting that

$$\mathbb{P}\left(|u^\top v| < \frac{\|v\|_2}{N^2}\sqrt{\frac{8}{\pi d_x}}\right) = \mathbb{P}\left(|u_1| < \frac{1}{N^2}\sqrt{\frac{8}{\pi d_x}}\right)$$

holds for any $v \in \mathbb{R}^{d_x}$ by the symmetry of the uniform distribution. Now we proceed as

$$\mathbb{P}\left(|u_1| < \frac{1}{N^2}\sqrt{\frac{8}{\pi d_x}}\right) = 2 \times \mathbb{P}\left(0 < u_1 < \frac{1}{N^2}\sqrt{\frac{8}{\pi d_x}}\right)$$

$$= \frac{2}{\texttt{Area}(\mathbb{S}^{d_x-1})} \times \int_{\arccos\left(\frac{1}{N^2}\sqrt{\frac{8}{\pi d_x}}\right)}^{\frac{\pi}{2}} \texttt{Area}(\mathbb{S}^{d_x-2}) \cdot (\sin\phi)^{d_x-2} \, \mathrm{d}\phi$$

$$= 2 \times \frac{\texttt{Area}(\mathbb{S}^{d_x-2})}{\texttt{Area}(\mathbb{S}^{d_x-1})} \times \int_{\arccos\left(\frac{1}{N^2}\sqrt{\frac{8}{\pi d_x}}\right)}^{\frac{\pi}{2}} (\sin\phi)^{d_x-2} \, \mathrm{d}\phi$$

$$= \frac{2}{\sqrt{\pi}} \times \frac{(d_x-1)\Gamma(\frac{d_x}{2}+1)}{d_x\Gamma(\frac{d_x}{2}+\frac{1}{2})} \times \int_{\arccos\left(\frac{1}{N^2}\sqrt{\frac{8}{\pi d_x}}\right)}^{\frac{\pi}{2}} (\sin\phi)^{d_x-2} \, \mathrm{d}\phi$$

$$< \sqrt{\frac{2}{\pi}} \times \frac{(d_x-1)\sqrt{d_x+2}}{d_x} \times \int_{\arccos\left(\frac{1}{N^2}\sqrt{\frac{8}{\pi d_x}}\right)}^{\frac{\pi}{2}} 1 \, \mathrm{d}\phi$$

$$\leq \sqrt{\frac{2d_x}{\pi}} \times \int_{\arccos\left(\frac{1}{N^2}\sqrt{\frac{8}{\pi d_x}}\right)}^{\frac{\pi}{2}} 1 \, \mathrm{d}\phi$$

$$= \sqrt{\frac{2d_x}{\pi}}\left(\frac{\pi}{2} - \arccos\left(\frac{1}{N^2}\sqrt{\frac{8}{\pi d_x}}\right)\right)$$

$$= \sqrt{\frac{2d_x}{\pi}} \arcsin\left(\frac{1}{N^2}\sqrt{\frac{8}{\pi d_x}}\right)$$

$$\leq \sqrt{\frac{2d_x}{\pi}} \times \frac{\pi}{2} \times \frac{1}{N^2}\sqrt{\frac{8}{\pi d_x}} = \frac{2}{N^2}$$

where $\texttt{Area}(\cdot)$ denotes the surface area of the object, i.e., $\texttt{Area}(\mathbb{S}^{d_x-1}) = d_x \pi^{\frac{d_x}{2}}/\Gamma(\frac{d_x}{2}+1)$. Here, the first inequality follows from the Gautschi's inequality (see Lemma 18) and $\sin \pi \leq 1$. The second inequality follows from the fact that $\frac{(t-1)\sqrt{t+2}}{t} \leq \sqrt{t}$ for $t \geq 1$. The third inequality follows from the fact that $\phi \leq \frac{\pi}{2} \sin \phi$ for any $0 \leq \phi \leq \frac{\pi}{2}$. This completes the proof of Lemma 13.

Lemma 18 (Gautschi's inequality (Gautschi, 1959)) *For any $x > 0, s \in (0, 1)$,*

$$x^{1-s} < \frac{\Gamma(x+1)}{\Gamma(x+s)} < (x+1)^{1-s}.$$

F.2. Proof of Lemma 14

In this proof, we assume that $K > N^2/4$ and $T = \left\lceil \frac{K}{\lfloor (d+1)/2 \rfloor} \right\rceil$ since implementing the identity function is sufficient otherwise. To begin with, we first define a network $f_b(x) : [0, K) \to [0, T)$ for $b = (b_i)_{i=1}^{\lfloor (d-1)/2 \rfloor} \in [T]^{\lfloor (d-1)/2 \rfloor}$ as

$$
f_b(x) := \begin{cases} x & \text{if } x < T \\ (x + b_1) \bmod T & \text{if } T \leq x < 2T \\ (x + b_2) \bmod T & \text{if } 2T \leq x < 3T \\ \quad \vdots \\ (x + b_{\lfloor (d-1)/2 \rfloor}) \bmod T & \text{if } \lfloor \frac{d-1}{2} \rfloor T \leq x \end{cases}
$$

$$
= x - T \times \mathbf{1}[x \geq T] + \sum_{i=1}^{\lfloor (d-1)/2 \rfloor} \left(\left(b_i - \sum_{j=1}^{i-1} b_j \right) \times \mathbf{1}[x \geq iT] - T \times \mathbf{1}[x + b_i \geq (i+1)T] \right). \tag{5}
$$

One can easily observe that f_b can be implemented by a STEP + ID network of 1 hidden layer and width d as $T \times \mathbf{1}[x \geq T]$ in (5) can be absorbed into $\left(b_i - \sum_{j=1}^{i-1} b_j \right) \times \mathbf{1}[x \geq iT]$ in (5) for $i = 1$.

Now, we show that if $T > N^2/4$, then there exist $b \in [T]^{\lfloor (d-1)/2 \rfloor}$ such that $\left| \lfloor f_b(\mathcal{X}) \rfloor \right| = N$ to complete the proof. Our proof utilizes the mathematical induction on i: If there exist $b_1, \ldots, b_{i-1} \in [T]$ such that

$$
\lfloor f_b(\{x \in \mathcal{X} : x < iT\}) \rfloor = \binom{[T]}{\left| \lfloor \{x \in \mathcal{X} : x < iT\} \rfloor \right|}, \tag{6}
$$

then there exists $b_i \in [T]$ such that

$$
\lfloor f_b(\{x \in \mathcal{X} : x < (i+1)T\}) \rfloor = \binom{[T]}{\left| \lfloor \{x \in \mathcal{X} : x < (i+1)T\} \rfloor \right|}. \tag{7}
$$

Here, one can observe that the statement trivially holds for the base case, i.e., for $\{x \in \mathcal{X} : x < T\}$. Now, using the induction hypothesis, suppose that there exist $b_1, \ldots, b_{i-1} \in [T]$ satisfying (6). Now, we prove that there exists $b_i \in [T]$ such that $\mathcal{S}_{b_i} := \lfloor f_b(\{x \in \mathcal{X} : iT \leq x < (i+1)T\}) \rfloor$ does not intersect with $\mathcal{T} := \lfloor f_b(\{x \in \mathcal{X} : x < iT\}) \rfloor$, i.e, (7) holds. Consider the following inequality:

$$
\sum_{b_i \in [T]} |\mathcal{S}_{b_i} \cap \mathcal{T}| = |\mathcal{S}_{b_i}| \times |\mathcal{T}| \leq \frac{N^2}{4}
$$

where the equality follows from the fact that for each $x \in \{x \in \mathcal{X} : iT \leq x < (i+1)T\}$, there exists exactly $|\mathcal{T}|$ values of b_i so that $\lfloor x \bmod T \rfloor \in \mathcal{T}$. However, since the number of possible choices of b_i is T, if $T > N^2/4$, then there exists $b_i \in [T]$ such that $\mathcal{S}_{b_i} \cap \mathcal{T} = \emptyset$, i.e., (7) holds. This completes the proof of Lemma 14.

F.3. Proof of Lemma 15

In this proof, we assume that $K > N$ and $T = \max\left\{N \times \lceil N/C \rceil, \lceil K/2 \rceil\right\}$ since implementing the identity function is sufficient otherwise. The proof of Lemma 15 is similar to that of Lemma 14. To begin with, we first define a network $f_b(x) : [0, K) \to [0, T)$ for $b = (b_i)_{i=1}^C \in [T]^C$ and $T =: M_1 < M_2 < \cdots < M_{C+1} := K$ so that $|\{x \in \mathcal{X} : M_i \leq x < M_{i+1}\}| \leq \lceil \frac{N}{C} \rceil \leq \lfloor \frac{T}{N} \rfloor$ for all i as

$$
f_b(x) := \begin{cases}
x & \text{if } x < T \\
(x + b_1) \bmod T & \text{if } M_1 = T \leq x < M_2 \\
(x + b_2) \bmod T & \text{if } M_2 \leq x < M_3 \\
\quad \vdots \\
(x + b_C) \bmod T & \text{if } M_C \leq x
\end{cases}
$$

$$
= x - 2T \times \mathbf{1}[x \geq T]
$$
$$
+ \sum_{i=1}^C \left(\left(T + b_i - \sum_{j=1}^{i-1} b_j \right) \times \mathbf{1}[x \geq M_i] - T \times \mathbf{1}\left[x \geq \min\{2T - b_i, M_{i+1}\} \right] \right) \quad (8)
$$

where (8) holds as $T \geq \lceil \frac{K}{2} \rceil$. Here, one can easily implement f_b by a STEP + ID network of L hidden layer and d_ℓ neurons at the ℓ-th hidden layer by utilizing one neuron for storing the input x, another one neuron for storing the temporary output, and other neurons implement indicator functions at each layer. This is because one do not require to store x in the last hidden layer and there exists $2C$ indicator functions to implement ($T \times \mathbf{1}[x \geq T]$ in (8) can be absorbed into $\left(T + b_i - \sum_{j=1}^{i-1} b_j \right) \times \mathbf{1}[x \geq M_i]$ in (8) for $i = 1$).

Now, we show that if $T \geq N$, then there exist $b \in [T]^C$ such that $\left| \lfloor f_b(\mathcal{X}) \rfloor \right| = N$ to completes the proof. Our proof utilizes the mathematical induction on i: If there exist $b_1, \ldots, b_{i-1} \in [T]$ such that

$$
\lfloor f_b(\{x \in \mathcal{X} : x < M_i\}) \rfloor = \binom{[T]}{|\lfloor \{x \in \mathcal{X} : x < M_i\} \rfloor|}, \quad (9)
$$

then there exists $b_i \in [T]$ such that

$$
\lfloor f_b(\{x \in \mathcal{X} : x < M_{i+1}\}) \rfloor = \binom{[T]}{|\lfloor \{x \in \mathcal{X} : x < M_{i+1}\} \rfloor|}. \quad (10)
$$

Here, one can observe that the statement trivially holds for the base case, i.e., for $\{x \in \mathcal{X} : x < T\}$. From the induction hypothesis, suppose that there exist $b_1, \ldots, b_{i-1} \in [T]$ satisfying (9). Now, we prove that there exists $b_i \in [T]$ such that $\mathcal{S}_{b_i} := \lfloor f_b(\{x \in \mathcal{X} : M_i \leq x < M_{i+1}\}) \rfloor$ does not intersect with $\mathcal{T} := \lfloor f_b(\{x \in \mathcal{X} : x \leq M_i\}) \rfloor$, i.e., (10) holds. Consider the following inequality:

$$
\sum_{b_i \in [T]} |\mathcal{S}_{b_i} \cap \mathcal{T}| \leq \lfloor \tfrac{T}{N} \rfloor \times \left(N - \lfloor \tfrac{T}{N} \rfloor \right) < T
$$

where the first inequality follows from the fact that $|\mathcal{S}_{b_i}| \leq \lfloor \frac{T}{N} \rfloor$, $|\mathcal{T}| \leq \left(N - \lfloor \frac{T}{N} \rfloor \right)$, and for each $x \in \{x \in \mathcal{X} : M_i \leq x < M_{i+1}\}$, there exists exactly $|\mathcal{T}|$ values of b_i so that $\lfloor x \bmod T \rfloor \in \mathcal{T}$. However, since the number of possible choices of b_i is T, there exists $b_i \in [T]$ such that $\mathcal{S}_{b_i} \cap \mathcal{T} = \emptyset$, i.e, (10) holds. This completes the proof of Lemma 15.

F.4. Proof of Lemma 16

In this proof, we explicitly construct f_θ satisfying the desired property stated in Lemma 16. To begin with, we describe the high-level idea of the construction. First, we construct a map $g : x \mapsto \left(\left\lfloor \frac{x}{B} \right\rfloor, x \bmod B \right)$ to transform an input $x \in [0, K)$ to a pair (a, b) such that $a \in [A]$ and $\lfloor b \rfloor \in [B]$. Here, we give labels to the pair (a, b) corresponding to the input x as $y(a, \lfloor b \rfloor) := y(\lfloor x \rfloor)$. Note that this label is well-defined as if $\lfloor x_1 \rfloor \neq \lfloor x_2 \rfloor$, then $\lfloor g(x_1) \rfloor \neq \lfloor g(x_2) \rfloor$. Now, we construct parameters $w_0, \ldots, w_{A-1}$ containing the label information of $\mathcal{X}$ as

$$w_a := \sum_{c \in [B]} y(a, c) \times 2^{-(c+1)D}, \tag{11}$$

i.e., from the $(\lfloor b \rfloor \cdot D + 1)$-th bit to the $(\lfloor b \rfloor \cdot D + D)$-th bit of the a-th parameter (w_a) contains the label information of $(a, \lfloor b \rfloor)$. Under this construction, we recover the label of $x \in \mathcal{X}$ by first mapping x to the pair (a, b) and extracting the parameter w_a. Then, we extract bits from the $(\lfloor b \rfloor \cdot D + 1)$-th bit to the $(\lfloor b \rfloor \cdot D + D)$-th bit of w_a to recover the label $y(x)$.

Now, we explicitly construct the network mapping $x \in \mathcal{X}$ to $y(x)$. To this end, we introduce the following lemma. The proof of Lemma 19 is presented in Appendix F.5.

Lemma 19 *Suppose that $A, B, K, L, d_1, \ldots d_L \in \mathbb{N}$ satisfy $AB \geq K$, $d_\ell \geq 2$ for all ℓ, and $\sum_{\ell=1}^{L} (d_\ell - 2) \geq A$. Then, for any finite set $\mathcal{X} \subset [0, K)$ and $w_0, \ldots, w_{A-1} \in \mathbb{R}$, there exists a* STEP $+$ ID *network f of L layers and d_ℓ neurons at the ℓ-th layer such that $f(x) = (w_{\lfloor x/B \rfloor}, x \bmod B)$ for all $x \in \mathcal{X}$.*

From Lemma 19, a STEP $+$ ID network of 1 hidden layer consisting of $A + 2$ hidden neurons can map $x \in \mathcal{X}$ to $(w_{\lfloor x/B \rfloor}, x \bmod B)$. Note that this network requires overall $4A + 10$ parameters ($3A + 6$ edges and $A + 4$ biases).

Finally, we introduce the following lemma for extracting from the $(\lfloor b \rfloor \cdot D + 1)$-th bit to the $(\lfloor b \rfloor \cdot D + D)$-th bit of $w_a (= w_{\lfloor x/B \rfloor})$. The proof of Lemma 20 is presented in Appendix F.6.

Lemma 20 *For any $D, B, R \in \mathbb{N}$, for any finite set $\mathcal{X} \subset [0, B)$, there exists a* STEP $+$ ID *network f of $2\lceil \frac{BD}{R} \rceil$ hidden layers and $\left((2R + 5)2^R + 2R^2 + 8R + 7 \right) \lceil \frac{BD}{R} \rceil - R2^R - R^2 + 3$ parameters satisfying the following property: For any $w = \sum_{i=1}^{BD} u_i \times 2^{-i}$ such that $u_i \in \{0, 1\}$, $f(x, w) = \sum_{i=1}^{D} u_{\lfloor x \rfloor \cdot D + i} \times 2^{D-i}$ for all $x \in \mathcal{X}$.*

From Lemma 20, a STEP $+$ ID network of $2\lceil \frac{BD}{R} \rceil$ hidden layers and $\left((2R + 5)2^R + 2R^2 + 8R + 7 \right) \lceil \frac{BD}{R} \rceil - R2^R - R^2 + 3$ parameters. Hence, by combining Lemma 19 and Lemma 20, f_θ can be implemented by a STEP $+$ ID network of $2\lceil \frac{BD}{R} \rceil + 2$ hidden layers and $4A + \left((2R+5)2^R + 2R^2 + 8R + 7 \right) \lceil \frac{BD}{R} \rceil - R2^R - R^2 + 3$ parameters. This completes the proof of Lemma 16.

F.5. Proof of Lemma 19

We design f as $f := f_L \circ \cdots \circ f_1(0, x)$ where each f_ℓ represents the function of the ℓ-th layer consisting of d_ℓ neurons. In particular, we construct f_ℓ as follows:

$$f_\ell(w, x) := \left(w + w_0 \times \mathbf{1}[\ell = 1] + \sum_{i=1}^{d_\ell - 2} (w_{c_\ell + i} - w_{c_\ell + i - 1}) \times \mathbf{1}[x \geq iB], \right.$$
$$\left. x - \sum_{i=1}^{d_\ell - 2} B \times \mathbf{1}[x \geq iB] \right)$$

where $c_\ell := \sum_{i=1}^{\ell-1} (d_\ell - 2)$. Then, f is the desired function and each f_ℓ can be implemented by a STEP + ID networks of 1 hidden layer consisting of d_ℓ hidden neurons (two neurons for storing x, w and other neurons are for $d_\ell - 2$ indicator functions). This completes the proof of Lemma 19.

F.6. Proof of Lemma 20

We construct $f(x, w) := 2^{R\lceil BD/R\rceil + D} \times f_{\lceil BD/R\rceil} \circ \cdots \circ f_1(x, w)$ where f_ℓ is defined as

$$
f_\ell(x, v) := \begin{cases}
\Big(x, v - \sum_{i=1}^R u_{(\ell-1)R+i} \times 2^{-(\ell-1)R-i} \\
\quad + \sum_{i=1}^R \big(u_{(\ell-1)R+i} \wedge \mathbf{1}[m_{i,\ell} \le x < m_{i,\ell} + 1]\big) \times 2^{-r_{i,\ell}}\Big) & \text{if } \ell < \lceil \frac{BD}{R} \rceil \\
v - \sum_{i=1}^R u_{(\ell-1)R+i} \times 2^{-(\ell-1)R-i} \\
\quad + \sum_{i=1}^R \big(u_{(\ell-1)R+i} \wedge \mathbf{1}[m_{i,\ell} \le x < m_{i,\ell} + 1]\big) \times 2^{-r_{i,\ell}} & \text{if } \ell = \lceil \frac{BD}{R} \rceil
\end{cases}.
$$

where u_i denotes the i-th bit of w in the binary representation, $\wedge$ denotes the binary 'and' operation, and $m_{i,\ell}, r_{i,\ell}$ are defined as

$$
m_{i,\ell} := \left\lfloor \frac{(\ell-1)R + i - 1}{D} \right\rfloor
$$

$$
r_{i,\ell} := \left\lceil \frac{BD}{R} \right\rceil R + \big((\ell-1)R + i - 1 \bmod D\big) + 1.
$$

Namely, each f_ℓ extracts R bits from the input w and it store the extracted bits to the last bits of v if the extracted bits are in from $(\lfloor x \rfloor \cdot D + 1)$-th bit to the $(\lfloor x \rfloor \cdot D + D)$-th bit of w. Thus, $f(x, w)$ is the desired function for Lemma 20.

To implement each f_ℓ by a STEP + ID network, we introduce Lemma 21. Note that we extract u_i from w in Lemma 21, i.e., we do not assume that u_i is given. From Lemma 21, a STEP + ID network of $2\lceil \frac{BD}{R} \rceil$ hidden layers consisting of $2^R + R + 1$ and $R + 2$ hidden neurons alternatively can map (x, w) to $\sum_{i=1}^D u_{\lfloor x \rfloor \cdot D + i} \times 2^{D-i}$ for all $x \in \mathcal{X}$. By considering the input dimension 2 and the output dimension 1, this network requires $\big((2R+5)2^R + 2R^2 + 8R + 7\big)\lceil \frac{BD}{R} \rceil - R2^R - R^2 + 3$ parameters $(\big((2R+4)2^R + 2R^2 + 6R + 4\big)\lceil \frac{BD}{R} \rceil - R2^R - R^2 + 2$ edges and $(2^R + 2R + 3)\lceil \frac{BD}{R} \rceil + 1$ biases). This completes the proof of Lemma 20.

Lemma 21 *A* STEP + ID *network of 2 hidden layers having $2^R + R + 1$ and $R + 2$ hidden neurons at the first and the second hidden layer, respectively, can implement f_ℓ.*

Proof We construct $f_\ell := g_3 \circ (g_2 \oplus g_1)$ where $g_2 \oplus g_1(x, v) := (g_2(x, v), g_1(x, v))$, i.e., the function concatenating the outputs of g_1, g_2. In this proof, we mainly focus on constructing f_ℓ for $\ell < \lceil \frac{BD}{R} \rceil$ since $f_{\lceil BD/R\rceil}$ can be implemented similarly. We define g_1, g_2, g_3 as

$$
g_1(x, v) := \bigg(x, v, \sum_{i=0}^{2^R-1} \eta_{1,i} \times \mathbf{1}[i \times 2^{-\ell R} \le v < (i+1) \times 2^{-\ell R}],
$$

$$
\vdots,
$$

$$
\sum_{i=0}^{2^R-1} \eta_{R,i} \times \mathbf{1}[i \times 2^{-\ell R} \le v < (i+1) \times 2^{-\ell R}]\bigg)
$$

$$
= (x, v, u_{(\ell-1)R+1}, \ldots, u_{\ell R})
$$

$$
g_2(x, v) := \Big(\mathbf{1}[m_{1,\ell} \le x < m_{1,\ell} + 1], \ldots, \mathbf{1}[m_{R,\ell} \le x < m_{R,\ell} + 1]\Big)
$$

$$g_3 \circ (g_1 \oplus g_2) := \left(x, v - \sum_{i=1}^{R} u_{(\ell-1)R+i} \times 2^{-(\ell-1)R-i} \right.$$
$$\left. + \sum_{i=1}^{R} (u_{(\ell-1)R+i} \wedge \mathbf{1}[m_{i,\ell} \leq x < m_{i,\ell} + 1]) \times 2^{-r_{i,\ell}} \right)$$
$$= \left(x, v - \sum_{i=1}^{R} u_{(\ell-1)R+i} \times 2^{-(\ell-1)R-i} \right.$$
$$\left. + \sum_{i=1}^{R} \mathbf{1}\left[u_{(\ell-1)R+i} + \mathbf{1}[m_{i,\ell} \leq x < m_{i,\ell} + 1] \geq 2 \right] \times 2^{-r_{i,\ell}} \right).$$

where $\eta_{r,i}$ is a constant such that $\eta_{r,i} = 1$ if $i \times 2^{-\ell R} \leq x < (i+1) \times 2^{-\ell R}$ implies that the $((\ell-1)R+r)$-th bit of x is 1 and $\eta_{r,i} = 0$ otherwise. Here, one can easily observe that g_1 can be implemented by a linear combinations of $\mathbf{1}[v \geq 2^{-\ell R}], \ldots, \mathbf{1}[v \geq (2^R - 1) \times 2^{-\ell R}]$ as it trivially holds that $\mathbf{1}[v \geq 0]$ and $\mathbf{1}[v < 2^{-(\ell-1)R}]$, i.e., $2^R - 1$ indicator functions are enough for g_1. Hence, g_1 can be implemented by a STEP + ID network of 1 hidden layer consisting of $2^R + 1$ hidden neurons where additional 2 neurons are for passing x, v. In addition, g_2 can be implemented by a STEP + ID network of R hidden neurons. Finally, g_3 can be implemented by a STEP + ID network of 1 hidden layer consisting of $R + 2$ hidden neurons (R neurons for R indicator functions and 2 neurons for passing x, v).

Therefore, f_ℓ can be implemented by a STEP + ID network of 2 hidden layers consisting of $2^R + R + 1$ hidden neurons for the first hidden layer and $R + 2$ hidden neurons for the second hidden layer. Note that implementation within two hidden layer is possible since the outputs of g_1, g_2 are simply linear combination of their hidden activation values and hence, can be absorbed into the linear map between hidden layers. This completes the proof of Lemma 21. ∎

F.7. Proof of Lemma 17

The main idea of the proof of Lemma 17 is identical to that of Lemma 16. Recall A, B and $w_0, \ldots, w_{A-1} \in \mathbb{R}$ from the proof of Lemma 16. Then, Lemma 17 is a direct corollary of Lemma 19 and Lemma 22.

Lemma 22 *For any $D, B \in \mathbb{N}$, for any finite set $\mathcal{X} \subset [0, B)$, for any $w = \sum_{i=1}^{DB} u_i \times 2^{-i}$ for some $u_i \in \{0, 1\}$, there exists a* STEP + ID *network f of $(2D + 1)B$ hidden layers and width 3 such that $f(x, w) = \sum_{i=1}^{D} u_{\lfloor x \rfloor \cdot D + i} \times 2^{-i}$ for all $x \in \mathcal{X}$.*

Proof We construct $f(x, w) := 2^{D(B+1)} \times f_{DB} \circ g_{DB} \circ h_{DB} \circ \cdots \circ f_1 \circ g_1 \circ h_1$ where f_ℓ, g_ℓ, h_ℓ are defined as

$$
\begin{aligned}
h_\ell(x, w) &:= \begin{cases} (x, w) & \text{if } \ell \bmod D \neq 1 \\ (x - 1 + 3K \times \mathbf{1}[x < 0], w) & \text{if } \ell \bmod D = 1 \end{cases} \\
g_\ell(x, w) &:= \left(x, w - 2^{-\ell} \times \mathbf{1}[w \geq 2^{-\ell}], \mathbf{1}[w \geq 2^{-\ell}]\right) \\
f_\ell(x, w, n) &= \left(x, w + 2^{-r_\ell} \times \mathbf{1}[x - n < -1]\right)
\end{aligned}
$$

where $r_\ell := DK + (\ell - 1 \mod D) + 1$. Now, we explain the constructions of f_ℓ, g_ℓ, h_ℓ. Let $x \in [0, B)$ and $w \in [0, 1)$ be inputs of f, i.e., consider $f(x, w)$. First, the indicator function in h_ℓ is activated only at $\ell = (\lfloor x \rfloor + 1) \cdot D + 1$ as $x < B$. In particular, the first entry of the output of $h_{(\lfloor x \rfloor + 1) \cdot D + 1}$ is greater than $2B$ and this is the maximum value of the first entry of the output of $h_{(\lfloor x \rfloor + 1) \cdot D + 1}$ as it monotonically decreases as ℓ grows. The indicator function in g_ℓ extracts and outputs the ℓ-th bit of w. Lastly, f_ℓ add the ℓ-th bit of w extracted by g_ℓ to the $\left(DK + (\ell - 1 \mod D) + 1\right)$-th bit of w if and only if $\ell \in \{\lfloor x \rfloor \cdot D + 1, \ldots, (\lfloor x \rfloor + 1) \cdot D\}$. This is because $x \in [-1, 0)$ if and only if $\ell \in \{\lfloor x \rfloor \cdot D + 1, \ldots, (\lfloor x \rfloor + 1) \cdot D\}$.

Here, h_ℓ at $\ell \bmod D = 1$ can be implemented by a STEP + ID network of 1 hidden layer and width 3, g_ℓ can be implemented by a STEP + ID network of 1 hidden layer and width 3, and f_ℓ can be implemented by a STEP + ID network of 1 hidden layer and width 3. Hence, f can be implemented by a STEP + ID network of $(2D + 1)B$ hidden layers and width 3. This completes the proof of Lemma 22. ∎