Finding Nearly Everything within Random Binary Networks

Kartik Sreenivasan **Shashank Rajput** **Jy-yong Sohn** **Dimitris Papailiopoulos**
University of Wisconsin-Madison

Abstract

A recent work by Ramanujan et al. (2020) provides significant empirical evidence that sufficiently overparameterized, random neural networks contain untrained subnetworks that achieve state-of-the-art accuracy on several predictive tasks. A follow-up line of theoretical work provides justification of these findings by proving that slightly overparameterized neural networks, with commonly used continuous-valued random initializations can indeed be pruned to approximate any target network. In this work, we show that the amplitude of those random weights does not even matter. We prove that any target network of width d and depth l can be approximated up to arbitrary accuracy ε by simply pruning a random network of binary $\{\pm 1\}$ weights that is wider and deeper than the target network only by a polylogarithmic factor of d, l and ε.

1 Introduction

As the number of parameters of state-of-the-art networks continues to increase, pruning has become a prime choice for sparsifying and compressing a model. A rich and long body of research, dating back to the 80s, shows that one can prune most networks to a tiny fraction of their size, while maintaining high accuracy (Mozer and Smolensky, 1989; Hassibi and Stork, 1993; Levin et al., 1994; LeCun et al., 1990; Han et al., 2015b,a; Li et al., 2016; Wen et al., 2016; Hubara et al., 2016, 2017; He et al., 2017; Wu et al., 2016; Zhu et al., 2016; He et al., 2018; Zhu and Gupta, 2017; Cheng et al., 2019; Blalock et al., 2020; Deng et al., 2020).

A downside of most of the classic pruning approaches is that they sparsify a model once it is trained to full accuracy, followed by significant fine-tuning, resulting in a computationally burdensome procedure. Frankle and Carbin (2018) conjectured the existence of *lottery tickets*, *i.e.*, sparse subnetworks at (or near) initialization, that can be trained—just once—to reach the accuracy of state-of-the-art dense models. This may help alleviate the computational burden of prior approaches, as training is predominantly carried on a much sparser model. The conjectured existence of these lucky tickets is referred to as the Lottery Ticket Hypothesis (LTH). Frankle and Carbin (2018) and Frankle et al. (2020) show that not only do *lottery tickets exist*, but also that the cost of "*winning the lottery*" is not very high.

Along the LTH literature, a curious phenomenon was observed; even at initialization and in the *complete absence* of training, one can find sub-networks of the random initial model that have prediction accuracy far beyond random guessing (Zhou et al., 2019; Ramanujan et al., 2020; Wang et al., 2019). Ramanujan et al. (2020) reported this in its most striking form: state-of-the-art accuracy models for CIFAR10 and ImageNet, *simply reside* within slightly larger, yet completely random networks, and appropriate pruning—and mere pruning—can reveal them! This "*pruning is all you need*" phenomenon is sometimes referred to as the Strong Lottery Ticket Hypothesis.

A recent line of work attempts to establish the theoretical validity of the Strong LTH by studying the following non-algorithmic question:

Can a random network be pruned to approximate a target function $f(x)$?

Here, f represents a bounded range labeling function that acts on inputs $x \in \mathcal{X}$, and is itself a neural network of finite width and depth. This assumption is not limiting, as neural networks are universal approximators (Stinchombe, 1989; Barron, 1993; Scarselli and Tsoi, 1998; Klusowski and Barron, 2018; Perekrestenko et al., 2018; Hanin, 2019; Kidger and Lyons, 2020). Note that the answer to the above question is trivial if one does not constraint the size of the random initial network, for all interesting cases of "*random*". Indeed, if we start with an exponentially wider random neural network

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

compared to the one representing f, by sheer luck, one can always find weights, for each layer, near-identical to those of any target neural network that is f. Achieving this result with a constrained overparameterization, *i.e.*, the degree by which the random network to be pruned is wider/deeper than f, is precisely why this question is challenging.

Malach et al. (2020) were the first to prove that the Strong LTH is true, assuming polynomial-sized overparameterization. Specifically, under some mild assumptions, they showed that to approximate a target network of width d and depth l to within error ε, it suffices to prune a random network of width $\widetilde{\mathcal{O}}(d^2l^2/\varepsilon^2)$ and depth $2l$. Pensia et al. (2020) offered an exponentially tighter bound using a connection to the SubsetSum problem. They showed that to approximate a target network within error ε, it is sufficient to prune a randomly initialized network of width $\mathcal{O}(d\log(dl/\varepsilon))$ and depth $2l$. A corresponding lower bound for constant depth networks was also established. Orseau et al. (2020) were also able to reduce the dependence on ε to logarithmic. They show that in order to approximate a target network within error ε, it suffices to prune a random network of width $\mathcal{O}(d^2\log(dl/\varepsilon))$ if the weights are initialized with the hyperbolic distribution. However, this bound on overparamaterization is still polynomial in the width d.

The above theoretical studies have focused exclusively on continuous distribution for initialization. However, in the experimental work by Ramanujan et al. (2020), the authors manage to obtain the best performance by pruning networks of scaled, *binary weights*. Training binary networks has been studied extensively in the past (Courbariaux et al., 2015; Simons and Lee, 2019) as they are compute, memory and hardware efficient, though in many cases they suffer from significant loss of accuracy. The findings of Ramanujan et al. (2020) suggest that the accuracy loss may not be fundamental to networks of binary weights, when such networks are learned by pruning. Arguably, since "*carving out*" sub-networks of random models is expressive enough to approximate a target function, *e.g.*, according to (Pensia et al., 2020; Malach et al., 2020), one is posed to wonder about the importance of weights altogether. So perhaps, *binary weights is all you need.*

Diffenderfer and Kailkhura (2021) showed that indeed scaled binary networks can be pruned to approximate any target function. The required overparameterization is similar to that of Malach et al. (2020), *i.e.*, polynomial in the width, depth and error of approximation. In a similar vein to the improvement that Pensia et al. (2020) offered over the bounds of Malach et al. (2020), we explore whether such an improvement is possible on the results of Diffenderfer and Kailkhura (2021).

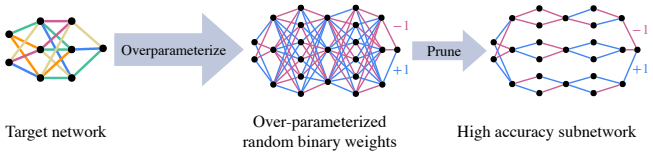


Figure 1: Approximating a target network with high accuracy by pruning overparameterized random binary network. In this paper, we show that logarithmic overparameterization in both width and depth is sufficient.

Our Contributions: In this work, we offer an exponential improvement to the theoretical bounds by Diffenderfer and Kailkhura (2021), establishing the following.

Theorem 1. *(informal) Consider a randomly initialized, FC, binary* $\{\pm 1\}$ *network of ReLU activations, with depth* $\Theta\left(l\log\left(\frac{dl}{\varepsilon}\right)\right)$ *and width* $\Theta\left(d\log^2\left(\frac{dl}{\varepsilon\delta}\right)\right)$*, with the last layer consisting of scaled binary weights* $\{\pm C\}$*. Then, there exists a constant* C *such that this network can be pruned to approximate* ***any*** *FC ReLU network, up to error* $\varepsilon > 0$ *with depth* l *and width* d*, with probability at least* $1-\delta$*.*

Therefore, we show that in order to approximate any target network, it suffices to just prune a logarithmically overparameterized binary network (Figure 1). In contrast to Diffenderfer and Kailkhura (2021), our construction only requires that the last layer be scaled while the rest of the network is purely binary $\{\pm 1\}$. We believe that the tightness of our bound comes from using depth more effectively. While Diffenderfer and Kailkhura (2021) use a construction that is only $2l$ in depth, it is wider than the target network by a polynomial factor. In contrast, our construction is both logarithmically wider *and* deeper than the target network thereby requiring exponentially smaller overaparameterization in terms of total parameters. We show a detailed comparison of the known Strong LTH results in Table 1.

We would also like to remark that our bounds more closely justify the experimental findings from Ramanujan et al. (2020) and Diffenderfer and Kailkhura (2021). Both works demonstrate that the amount of overparameterization required to find a high accuracy subnetwork is small, and surely not polynomial. Our bounds reflect this observation.

In light of our theoretical results, one may wonder why in the literature of training, *i.e.*, assigning a sign pattern to fixed architecture, binary networks, a loss of accuracy is observed, *e.g.*, Rastegari et al. (2016). Is this an algorithmic artifact, or does pruning random signs offer higher expressivity than assigning the signs? We show that there exist target functions that can be well approximated by pruning binary networks, yet none of

Reference	Width	Depth	Total Params	Weights
Malach et al. (2020)	$\widetilde{\mathcal{O}}(d^2l^2/\varepsilon^2)$	$2l$	$\widetilde{\mathcal{O}}(d^2l^3/\varepsilon^2)$	Real
Orseau et al. (2020)	$\mathcal{O}(d^2\log(dl/\varepsilon)$	$2l$	$\mathcal{O}(d^2l\log(dl/\varepsilon)$	Real (Hyperbolic)
Pensia et al. (2020)	$\mathcal{O}(d\log(dl/\min\{\varepsilon,\delta\})$	$2l$	$\mathcal{O}(dl\log(dl/\min\{\varepsilon,\delta\})$	Real
Diffenderfer and Kailkhura (2021)	$\mathcal{O}((ld^{3/2}/\varepsilon)+ld\log(ld/\delta))$	$2l$	$\mathcal{O}((l^2d^{3/2}/\varepsilon)+l^2d\log(ld/\delta))$	$\{\pm\varepsilon\}$
Ours, Theorem 1	$\mathcal{O}(d\log^2(dl/\varepsilon\delta))$	$\mathcal{O}(l\log(dl/\varepsilon))$	$\mathcal{O}(dl\log^3(dl/\varepsilon\delta))$	Binary-$\{\pm1\}$[1]

Table 1: Comparing the upper bounds for the overparameterization needed to approximate a target network (of width d and depth l) within error $\varepsilon>0$ with probability at least $1-\delta$ by pruning a randomly initialized network.

all possible, binary, fully-connected (FC) networks can approximate it.

Proposition 1. *(informal) There exist a function f that can be represented by pruning a random 2-layer binary network of width d, but not by any 2-layer fully-connected binary network of width d.*

Note that although finding a subnetwork of a random binary network results in a "ternary" architecture (e.g., 0 becomes a possible weight), the total number of possible choices of subnetworks is 2^N, if N is the total number of weights. This is equal to the total number of sign assignments of the same FC network. Yet, as shown in the proposition above, pruning a random FC network is provably more expressive than finding a sign assignment for the same architecture.

2 Preliminaries and Problem Setup

Let $f(\boldsymbol{x}):\mathbb{R}^{d_0}\to\mathbb{R}$ be the target FC network with l layers and ReLU activations, represented as

$$f(\boldsymbol{x})=\sigma(\boldsymbol{W}_l\sigma(\boldsymbol{W}_{l-1}\dots\sigma(\boldsymbol{W}_1\boldsymbol{x}))),$$

where $\boldsymbol{x}\in\mathbb{R}^{d_0}$ is the input, $\sigma(\boldsymbol{z})=\max\{\boldsymbol{z},0\}$ is the ReLU activation and $\boldsymbol{W}_i\in\mathbb{R}^{d_i\times d_{i-1}}$ is the weight matrix of layer $i\in[l]$. With slight abuse of terminology, we will refer to f as a network, as opposed to a labeling function. We then consider a binary[1] network of depth l'

$$g(\boldsymbol{x})=\sigma((\varepsilon'\boldsymbol{B}_{l'})\sigma(\boldsymbol{B}_{l'-1}\dots\sigma(\boldsymbol{B}_1\boldsymbol{x}))),$$

where $\boldsymbol{B}_i\in\{-1,+1\}^{d_i'\times d_{i-1}'}$ is a binary weight matrix, with all weights drawn uniformly at random from $\{\pm1\}$, for all layers $i\in[l']$ and the last layer is multiplied by a factor of $\varepsilon'>0$. The scaling factor is calculated precisely in Section 3.2.3, where we show that it is unavoidable for function approximation unless the problem is that of classification.

Our goal is to find the smallest network g so that it contains a subnetwork $\tilde{g}$ which approximates f closely. More precisely, we will bound the overparameterization of the binary network, under which one can find supermask matrices $\boldsymbol{M}_i\in\{0,1\}^{d_i'\times d_{i-1}'}$, for each layer $i\in[l']$, such that the pruned network

$$\tilde{g}(\boldsymbol{x})=\sigma(\varepsilon'(\boldsymbol{M}_{l'}\odot\boldsymbol{B}_{l'})\sigma((\boldsymbol{M}_{l'-1}\odot\boldsymbol{B}_{l'-1})\dots\dots\sigma((\boldsymbol{M}_1\odot\boldsymbol{B}_1)\boldsymbol{x})))$$

is ε-close to f in the sense of uniform approximation over the unit-ball, *i.e.*,

$$\max_{\boldsymbol{x}\in\mathbb{R}^{d_0}:||\boldsymbol{x}||\le1}||f(\boldsymbol{x})-\tilde{g}(\boldsymbol{x})||\le\varepsilon$$

for some desired $\varepsilon>0$. In this paper, we show g only needs to be polylogarithmically larger than the target network f to have this property. We formalize this and provide a proof in the following sections.

Henceforth, we denote $[k]=\{1,2,\cdots,k\}$ for some positive integer k. Unless otherwise specified, $||\cdot||$ refers to the ℓ_2 norm which induces the spectral norm for matrices. We also use the max norm of a matrix, defined as $||\boldsymbol{A}||_{\max}:=\max_{ij}|A_{ij}|$. The element-wise product between two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ is denoted by $\boldsymbol{A}\odot\boldsymbol{B}$. We assume without loss of generality that the weights are specified in the base-10 system. However, since we don't specify the base of the logarithm explicitly in our computations, we use the $\Theta(\cdot)$ notation to hide constant factors that may arise from choosing different bases.

3 Strong Lottery Tickets by Binary Expansion

In this section, we formally present our approximation results. We show that in order to approximate any target network $f(\boldsymbol{x})$ within arbitrary approximation error ε, it suffices to prune a random binary[1] network $g(\boldsymbol{x})$ that is just polylogarithmically deeper and wider than the target network.

3.1 Main Result

First, we point out that the scaling factor ε' in the final layer of $g(\boldsymbol{x})$ is necessary for achieving arbitrarily small approximation error for any target network $f(\boldsymbol{x})$. In other words, it is impossible to approximate an arbitrary target network with a purely binary $\{\pm1\}$ network regardless of the overparameterization. To see

[1]The weights of all the layers are purely binary $\{\pm1\}$ except for the last layer which is scaled so that it is $\{\pm\varepsilon'\}$ where $\varepsilon'=(\varepsilon/d^2l)^l$.

this, note that for the simple target function $f(x) = \varepsilon x,\ x \in [0,1]$ and $\varepsilon \in [0.5, 1)$, the best approximation possible by a binary network is $g(x) = x$ and therefore $\max_{x\in\mathbb{R}:|x|\leq 1} |f(x) - g(x)| \geq (1-\varepsilon)$ for any binary network g. We will show that just by allowing the weights of the final layer to be scaled, we can provide a uniform approximation guarantee while the rest of the network remains binary $\{\pm 1\}$. Formally, we have the following theorem:

Theorem 1. *Consider the set of FC ReLU networks $\mathcal{F}$ defined as*

$$\mathcal{F} = \{f : f(\boldsymbol{x}) = \sigma(\boldsymbol{W}_l\sigma(\boldsymbol{W}_{l-1}\dots\sigma(\boldsymbol{W}_1\boldsymbol{x}))), \\ \forall i\ \boldsymbol{W}_i \in \mathbb{R}^{d_i\times d_{i-1}}\ ||\boldsymbol{W}_i|| \leq 1\},$$

and let $d = \max_i d_i$. For any $\varepsilon > 0$, let $g(\boldsymbol{x}) = \sigma(\varepsilon'\boldsymbol{B}_{l'}\sigma(\boldsymbol{B}_{l'-1}\dots\sigma(\boldsymbol{B}_1\boldsymbol{x})))$ (here $\varepsilon' = (\varepsilon/d^2l)^l$) be a randomly initialized network with depth $l' = \Theta(l\log(d^2l/\varepsilon))$ such that every weight is drawn uniformly from $\{-1,+1\}$ and the layer widths are $\Theta\left(\log d^2l/\varepsilon \cdot \log\left(\frac{dl\log^2(d^2l/\varepsilon)}{\delta}\right)\right)$ times wider than $f(\boldsymbol{x})$.

Then, with probability at least $1-\delta$, for every $f \in \mathcal{F}$, there exist pruning matrices $\boldsymbol{M}_i$ such that

$$\max_{\boldsymbol{x}\in\mathbb{R}^{d_0}:||\boldsymbol{x}||\leq 1} |f(\boldsymbol{x}) - \tilde{g}(\boldsymbol{x})| \leq \varepsilon$$

holds where

$$\tilde{g}(\boldsymbol{x}) := \sigma(\varepsilon'(\boldsymbol{M}_{l'}\odot\boldsymbol{B}_{l'})\sigma((\boldsymbol{M}_{l'-1}\odot\boldsymbol{B}_{l'-1})\dots \\ \dots\sigma((\boldsymbol{M}_1\odot\boldsymbol{B}_1)\boldsymbol{x}))).$$

Remark 1. *The dimensions of the weight matrices of $g(\boldsymbol{x})$ in Theorem 1 are specified more precisely below. Let $p = (d^2l/\varepsilon)$. Since $l' = l\log(p)$, we have $\lfloor\log(p)\rfloor$ layers in $g(\boldsymbol{x})$ that approximates each layer in $f(\boldsymbol{x})$. For each $i \in [l]$, the dimension of $\boldsymbol{B}_{(i-1)\lfloor\log(p)\rfloor+1}$ is*

$$\Theta\left(d_{i-1}\log(p)\log\left(\frac{dl\log^2(p)}{\delta}\right)\right)\times d_{i-1},$$

the dimension of $\boldsymbol{B}_{i\lfloor\log(p)\rfloor}$ is

$$d_i \times \Theta\left(d_{i-1}\log(p)\log\left(\frac{dl\log^2(p)}{\delta}\right)\right)$$

and the remaining $\boldsymbol{B}_{(i-1)\lfloor\log(p)\rfloor+k}$ where $1 < k < \lfloor\log(p)\rfloor$ have the dimension

$$\Theta\left(d_{i-1}\log(p)\log\left(\frac{dl\log^2(p)}{\delta}\right)\right) \\ \times\Theta\left(d_{i-1}\log(p)\log\left(\frac{dl\log^2 p}{\delta}\right)\right).$$

3.2 Proof of Theorem 1

First, we show in Section 3.2.1 that any target network in $f(\boldsymbol{x}) \in \mathcal{F}$ can be approximated within $\varepsilon > 0$, by another network $\hat{g}_p(\boldsymbol{x})$ having weights of finite-precision at most p digits where p is logarithmic in d, l, and ε.

Then, in Section 3.2.2, we show that any finite precision network can be represented exactly using a binary network where all the weights are binary $\{\pm 1\}$ except the last layer, and the last layer weights are scaled-binary $\{\pm\varepsilon'\}$. We do this by first showing that any finite-precision network is equivalent to a network with integer weights in every layer except the last using a simple scaling argument. We then prove Lemma 6 which shows the deterministic construction of a binary network using diamond-shaped gadgets that can be pruned to approximate any integer network. Theorem 2 extends the result to the case when the network is initialized with random binary weights.

Putting these together completes the proof of Theorem 1 as shown in Section 3.2.3.

3.2.1 Logarithmic precision is sufficient

First, we consider the simplest setting wherein the target network contains a single weight *i.e.*, $h(x) = \sigma(wx)$, where x, w are scalars, the absolute values of which are bounded by 1. This assumption can be relaxed to any finite norm bound. We begin by noting that $\log(1/\varepsilon)$ digits of precision are sufficient to approximate a real number within error ε, as formalized below

Fact 1. *Let $w \in \mathbb{R},\ |w| \leq 1$ and $\hat{w}$ be a finite-precision truncation of w with $\lceil\Theta(\log(1/\varepsilon))\rceil$ digits. Then $|w - \hat{w}| \leq \varepsilon$ holds.*

Now we state the result for the case when the target network contains a single weight w.

Lemma 1. *Consider a network $h(x) = \sigma(wx)$ where $w \in \mathbb{R}, |w| \leq 1$. For a given $\varepsilon > 0$, let $\hat{w}$ be a finite-precision truncation of w up to $\log(1/\varepsilon)$ digits and let $\hat{g}_{\log(1/\varepsilon)}(x) = \sigma(\hat{w}x)$. Then we have*

$$\max_{x\in\mathbb{R}:|x|\leq 1} |h(x) - \hat{g}_{\log(1/\varepsilon)}(x)| \leq \varepsilon.$$

Proof. By Fact 1, we know that $|w - \hat{w}| \leq \varepsilon$. Applying Cauchy-Schwarz with $|x| \leq 1$ gives us $|\hat{w}x - wx| \leq \varepsilon$. Since this holds for any x and ReLU is 1-Lipschitz, the result follows. □

Lemma 1 can be extended to show that it suffices to consider finite-precision truncation up to $\log(d^2l/\varepsilon)$ digits to approximate a network for width d and depth l. This is stated more formally below.

Lemma 2. *Consider a network* $h(\boldsymbol{x}) = \sigma(\boldsymbol{W}_l\sigma(\boldsymbol{W}_{l-1}\dots\sigma(\boldsymbol{W}_1\boldsymbol{x})))$ *where* $\boldsymbol{W}_i \in \mathbb{R}^{d_i\times d_{i-1}}$, $||\boldsymbol{W}_i|| \le 1$. *For a given* $\varepsilon > 0$, *define* $\hat{g}_{\log(d^2l/\varepsilon)}(\boldsymbol{x}) = \sigma(\widehat{\boldsymbol{W}}_l\sigma(\widehat{\boldsymbol{W}}_{l-1}\dots\sigma(\widehat{\boldsymbol{W}}_1\boldsymbol{x})))$ *where* $\widehat{\boldsymbol{W}}_i$ *is a finite precision truncation of* $\boldsymbol{W}_i$ *up to* $\log(d^2l/\varepsilon)$ *digits, where* $d = \max_i d_i$. *Then we have*

$$\max_{\boldsymbol{x}\in\mathbb{R}^{d_0}:||\boldsymbol{x}||\le 1} |h(\boldsymbol{x}) - \hat{g}_{\log(d^2l/\varepsilon)}(\boldsymbol{x})| \le \varepsilon.$$

We provide the proof of Lemma 2 as well as approximation results for a single neuron and layer in Appendix A.1.

3.2.2 Binary weights are sufficient

We begin by showing that any finite-precision FC ReLU network can be represented perfectly as a FC ReLU network with integer weights in every layer except the last, using a simple scaling argument. Since ReLU networks are positive homogenous, we have that $\sigma(c\cdot z) = c\cdot\sigma(z)$ for any $c > 0$. Given a network g_p where all the weights are of finite-precision at most p, we can apply this property layerwise with the scaling factor $c = 10^p$ so that,

$$\begin{aligned} f(\boldsymbol{x}) &= \sigma(\boldsymbol{W}_l\sigma(\boldsymbol{W}_{l-1}\dots\sigma(\boldsymbol{W}_1\boldsymbol{x}))) \\ &= \frac{1}{c^l}\sigma(c\boldsymbol{W}_l\sigma(c\boldsymbol{W}_{l-1}\dots\sigma(c\boldsymbol{W}_1\boldsymbol{x}))) \\ &= \sigma(c'\widehat{\boldsymbol{W}}_l\sigma(\widehat{\boldsymbol{W}}_{l-1}\dots\sigma(\widehat{\boldsymbol{W}}_1\boldsymbol{x}))) \end{aligned} \tag{1}$$

where $\widehat{\boldsymbol{W}}_i = 10^p\boldsymbol{W}_i$ is a matrix of integer weights and $c' = \frac{1}{c^l}$. Therefore, the rescaled network has integer weights in every layer except the last layer which has the weight matrix $c'\widehat{\boldsymbol{W}}_l = (c^{-l})\boldsymbol{W}_l$.

In the remaining part of this section, we show that any FC ReLU network with integer weights can be represented exactly by pruning a purely binary $\{\pm1\}$ FC ReLU network which is just polylogarithmic wider and deeper. More precisely, we prove the following result.

Theorem 2. *Consider the set of FC ReLU networks with integer weights* $\mathcal{F}_W$ *defined as*

$$\begin{aligned} \mathcal{F}_W = \{f : f(\boldsymbol{x}) = \sigma(\boldsymbol{W}_l\sigma(\boldsymbol{W}_{l-1}\dots\sigma(\boldsymbol{W}_1\boldsymbol{x}))), \\ \forall i\ \boldsymbol{W}_i \in \mathbb{Z}^{d_i\times d_{i-1}}\ ||\boldsymbol{W}_i||_{max} \le W\} \end{aligned}$$

where $W > 0$. *Define* $d = \max_i d_i$ *and let* $g(\boldsymbol{x}) = \sigma(\boldsymbol{B}_{l'}\sigma(\boldsymbol{B}_{l'-1}\dots\sigma(\boldsymbol{B}_1\boldsymbol{x})))$ *be a network with depth* $l' = \Theta(l\log(|W|))$ *where every weight is uniform-randomly generated from* $\{-1,+1\}$ *and the layer widths are* $\Theta\left(\log|W|\cdot\log\left(\frac{dl\log^2|W|}{\delta}\right)\right)$ *times wider than* $f(\boldsymbol{x})$.

Then, with probability at least $1-\delta$, *for every* $f\in\mathcal{F}$, *there exist pruning matrices* $\boldsymbol{M}_i$ *such that*

$$f(\boldsymbol{x}) = \tilde{g}(\boldsymbol{x})$$

holds for any $\boldsymbol{x}\in\mathbb{R}^{d_0}$ *where* $\tilde{g}(\boldsymbol{x}) := \sigma((\boldsymbol{M}_{l'}\odot\boldsymbol{B}_{l'})\sigma((\boldsymbol{M}_{l'-1}\odot\boldsymbol{B}_{l'-1})\dots\sigma((\boldsymbol{M}_1\odot\boldsymbol{B}_1)\boldsymbol{x})))$.

Remark 2. *The dimensions of the weight matrices of* $g(\boldsymbol{x})$ *in Theorem 2 are specified more precisely below. Note that we have* $\lfloor\log|W|\rfloor$ *layers in* $g(\boldsymbol{x})$ *that exactly represents each layer in* $f(\boldsymbol{x})$. *For each* $i\in[l]$, *the dimension of* $\boldsymbol{B}_{(i-1)\lfloor\log|W|\rfloor+1}$ *is*

$$\Theta\left(d_{i-1}\log|W|\log\left(\frac{dl\log^2|W|}{\delta}\right)\right)\times d_{i-1},$$

the dimension of $\boldsymbol{B}_{i\lfloor\log|W|\rfloor}$ *is*

$$d_i\times\Theta\left(d_{i-1}\log|W|\log\left(\frac{dl\log^2|W|}{\delta}\right)\right)$$

and the remaining $\boldsymbol{B}_{(i-1)\lfloor\log|W|\rfloor+k}$ *where* $1 < k < \lfloor\log|W|\rfloor$ *have the dimension*

$$\begin{aligned} &\Theta\left(d_{i-1}\log|W|\log\left(\frac{dl\log^2|W|}{\delta}\right)\right) \\ \times&\Theta\left(d_{i-1}\log|W|\log\left(\frac{dl\log^2|W|}{\delta}\right)\right) \end{aligned}$$

Remark 3. *Note that* $\tilde{g}(\boldsymbol{x})$ *is* ***exactly*** *equal to* $f(\boldsymbol{x})$. *Furthermore, like Pensia et al. (2020) we provide a uniform guarantee for* all *networks in* $\mathcal{F}$ *by pruning a single over-parameterized network.*

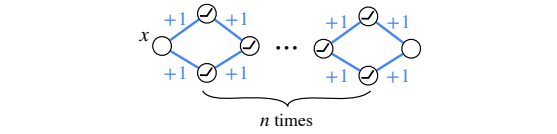
Remark 4. *Theorem 2 can be stated using a deterministic construction thereby avoiding the* $\log(1/\delta)$ *overparameterization. We extend to the random initialization setting by resampling this construction a sufficient number of times.*

Remark 5. *To resolve issues of numerical overflow, we can insert scaling neurons after every layer.*

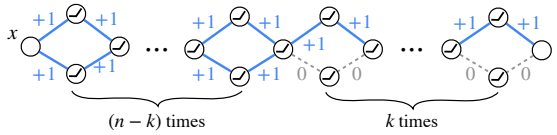
Remark 6. *The integer assumption can easily be converted to a finite-precision assumption using a simple scaling argument. Since all networks in practice use finite-precision arithmetic, Theorem 2 may be of independent interest to the reader. However, we emphasize here that there is no approximation error in this setting. Practitioners who are interested in small error*(10^{-k}) *can just apply Theorem 1 and incur an overparameterization factor of* $\mathcal{O}(k)$.

The proof of Theorem 2 will first involve a deterministic construction of a binary network that gives us the desired guarantee. The construction is based on a diamond-shaped gadget that allows us to approximate a single integer weight by pruning a binary ReLU network with just logarithmic overparameterization.

First, consider a target network that contains just a single integer weight *i.e.*, $h(x) = \sigma(wx)$. We will show that there exists a binary FC ReLU network $g(x)$ which can be pruned to approximate $h(x)$.



(a) $g_n(x) = 2^n \max\{0, x\}$



(b) $g_{n-k}(x) = 2^{n-k} \max\{0, x\}$

Figure 2: The diamond-shaped binary ReLU networks that compute $g_n(x)$ and $g_{n-k}(x)$, respectively. The dashed edges are just weights that have been "pruned" (set to 0). The output neuron is a simple linear activation.

Lemma 3. *Consider a network with a single weight, $h(x) = \sigma(wx)$ where $w \in \mathbb{Z}$. Then there exists a FC ReLU binary network $g(x)$ of width and depth $O(\log |w|)$ that can be pruned to $\tilde{g}(x)$ so that $\tilde{g}(x) = h(x)$ for all $x \in \mathbb{R}$.*

Proof. Note that since w is an integer, it can be represented using its binary (base-2) expansion

$$w = \text{sign}(w) \sum_{k=0}^{\lfloor \log_2 |w| \rfloor} z_k \cdot 2^k, \quad z_k \in \{0, 1\}. \tag{2}$$

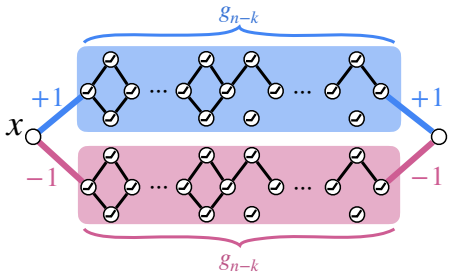
For ease of notation, we will use $\log(\cdot)$ to represent $\log_2(\cdot)$ going forward. Denote $n = \lfloor \log_2 |w| \rfloor$. The construction of $g_n(x)$ in Figure 2a shows that 2^n can be represented by a binary network with ReLU activations for any n. We will refer to this network as the diamond-shaped "gadget".

Note that the expansion in Equation (2) requires us to approximate 2^k for all $0 \le k \le n = \lfloor \log |w| \rfloor$. Luckily, any of these can be represented by just pruning $g_n(x)$ as shown in Figure 2b.

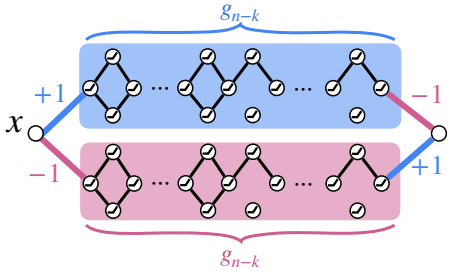
However, since our constructions use the ReLU activation, this only works when $x \ge 0$. Using the same trick as Pensia et al. (2020), we can extend this construction by mirroring it as shown in Figure 3. This gives us $f^+_{n-k}(x) := 2^{n-k}x$ and $f^-_{n-k}(x) := -2^{n-k}x$. The correctness of this mirroring trick relies on the simple observation that $wx = \sigma(wx) - \sigma(-wx)$.

Putting these together, we get $g_n(x) = \sigma(\pm \sum_{k=0}^{n} 2^k x)$ as shown in Figure 4. By pruning just the weights in the last layer, we can choose which terms to include. Setting $n = \lfloor \log |w| \rfloor$ completes the approximation.

To calculate the overparameterization required to approximate $h(x) = \sigma(wx)$, we simply count the parameters in the above construction. Each gadget g_k is a network of width 2 and depth $(\lfloor \log |w| \rfloor)$. To construct



(a) $f^+_{n-k}(x) = 2^{n-k}x$



(b) $f^-_{n-k}(x) = -2^{n-k}x$

Figure 3: We can use two instances of g_{n-k} to create (a) f^+_{n-k} and (b) f^-_{n-k} to approximate both positive weights and negative weights. The output neuron here has a linear activation.

f^+_k, we need two such gadgets. Therefore to construct f^+_k and f^-_k, we need width 4 and depth $(\lfloor \log |w| \rfloor)$. Repeating this for each $k \in 1, 2, \ldots, \lfloor \log |w| \rfloor$ shows that our construction is a network of width and depth $\mathcal{O}(\log |w|)$ which completes the proof of Lemma 3. □

Remark 7. *The network in Fig. 4 used for proving Lemma 3 can be written as*

$$g(x) = \sigma(\boldsymbol{M_v} \odot \boldsymbol{v})^T [(\boldsymbol{M}_n \odot \boldsymbol{B}_n)\sigma((\boldsymbol{M}_{n-1} \odot \boldsymbol{B}_{n-1}) \ldots \ldots \sigma((\boldsymbol{M}_1 \odot \boldsymbol{B}_1)\sigma((\boldsymbol{M_u} \odot \boldsymbol{u})x)))]),$$

where $\{\boldsymbol{M}_i\}_{i\in[n]}, \boldsymbol{M_v}, \boldsymbol{M_u}$ are mask matrices and $\{\boldsymbol{B}_i\}_{i\in[n]}, \boldsymbol{v}, \boldsymbol{u}$ are binary weight matrices. By pruning elements in $\boldsymbol{u}$ or $\boldsymbol{v}$, one can obtain $h(x) = \sigma(wx)$. We will always prune the last layer $\boldsymbol{v}$ as it makes the construction more efficient when we extend it to approximating a layer.

Now, we extend the construction to the case where the target function is a neural network with a single neuron *i.e.*, $h(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x})$.

Lemma 4. *Consider the set of single neuron networks $\mathcal{H}_{w_{max}} = \{h : h(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x}), \boldsymbol{w} \in \mathbb{Z}^d, ||\boldsymbol{w}||_\infty \le w_{max}\}$. Then there exists a FC ReLU binary network $g(\boldsymbol{x})$ of width $\mathcal{O}(d \log |w_{max}|)$ and depth $\mathcal{O}(\log |w_{max}|)$ that can be pruned to $\tilde{g}_h(\boldsymbol{x})$ for every $h \in \mathcal{H}_{w_{max}}$ so that $\tilde{g}_h(\boldsymbol{x}) = h(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^d$.*

Proof. A neuron can be written as $h(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x}) = \sigma(\sum_{i=1}^{d} w_i x_i)$. Therefore, we can just repeat our con-

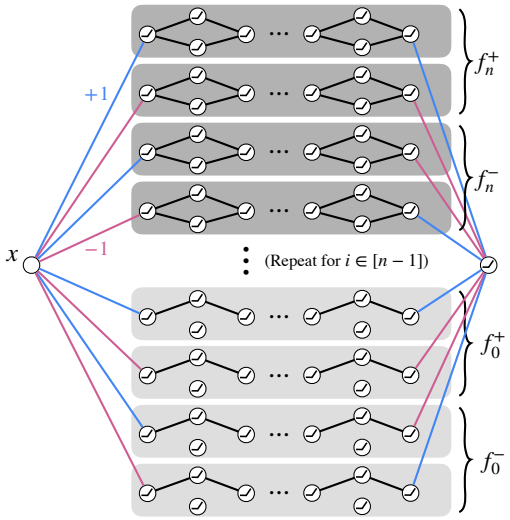


Figure 4: Illustration of $g_k(x) = \sum_{k=0}^{n}(f_k^+(x) + f_k^-(x)) = \sigma(\pm\sum_{k=0}^{n} 2^k x)$ which can be further pruned to approximate $f(x) = \sigma(wx)$ for any $w : |w| < 2^{n+1} - 1$

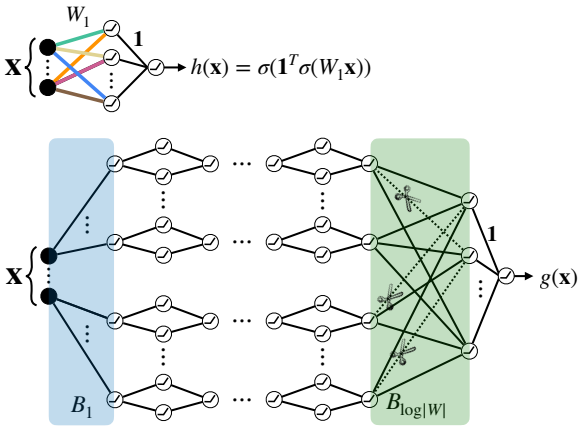


Figure 5: Illustration of the construction in Lemma 5: Approximating a 1-hidden layer network $h(\boldsymbol{x}) = \sigma(\mathbf{1}^T\sigma(\boldsymbol{W}_1\boldsymbol{x}))$ by pruning the appropriate binary network $g(\boldsymbol{x})$. Pruning the last layer allows us to "re-use" weights.

struction from above for each $w_i, i \in [d]$. This results in a network of width $\mathcal{O}(d \log |w_{max}|)$ while the depth remains unchanged at $\mathcal{O}(\log |w_{max}|)$. Note that the construction is identical for every $h \in \mathcal{H}_{w_{max}}$ since it only depends on the largest weight in the network. □

Next, we describe how to approximate a single layer target network and avoid a quadratic overparameterization.

Lemma 5. *Consider the set of single layer networks* $\mathcal{H}_W = \{h : h(\boldsymbol{x}) = \sigma(\mathbf{1}^T\sigma(\boldsymbol{W}_1\boldsymbol{x})),\ \boldsymbol{W}_1 \in \mathbb{Z}^{d_1 \times d_0},\ ||\boldsymbol{W}_1||_{max} \leq W\}$. *Then there exists a FC ReLU binary network* $g(\boldsymbol{x})$ *of width* $\mathcal{O}(\max\{d_0, d_1\} \log |W|)$ *and depth* $\mathcal{O}(\log |W|)$ *that can be pruned to* $\tilde{g}_h(\boldsymbol{x})$ *for every* $h \in \mathcal{H}$ *so that* $\tilde{g}_h(\boldsymbol{x}) = h(\boldsymbol{x})$ *for all* $\boldsymbol{x} \in \mathbb{R}^{d_0}$.

Proof. For ease of exposition, we describe the proof for a single h but it trivially applies to every $h \in \mathcal{H}_W$. Note that $\mathbf{1} \in \mathbb{R}^{d_1}$ is the vector of 1's. Naively extending the construction from Lemma 4 would require us to replace each of the d_0 neurons in the first layer by a network of width $\mathcal{O}(d_1 \log |W|)$ and depth $\mathcal{O}(\log |W|)$. This already needs a network of width $\mathcal{O}(d_0 d_1 \log |W|)$ which is a *quadratic* overparameterization. Instead, we take advantage of pruning only the last layer in the construction from Lemma 3 to re-use a sufficient number of weights and avoid the quadratic overparameterization. An illustration of this idea is shown in Figure 5. The key observation is that by pruning only the last layer of each $f_n(x)$ gadget, we leave it available to be re-used to approximate the weights of the remaining $(d_1 - 1)$ neurons. Therefore, the width of the network required to approximate $h(\boldsymbol{x})$ is just $\mathcal{O}(\max\{d_0, d_1\} \log |W|)$ and the depth remains $\mathcal{O}(\log |W|)$. □

Now we can tie it all together to show an approximation guarantee for any network in $\mathcal{F}_W$.

Lemma 6. *Consider the set of networks* $\mathcal{F}_W = \{f : f(\boldsymbol{x}) = \sigma(\boldsymbol{W}_l\sigma(\boldsymbol{W}_{l-1} \ldots \sigma(\boldsymbol{W}_1\boldsymbol{x}))),\ \boldsymbol{W}_i \in \mathbb{Z}^{d_i \times d_{i-1}},\ ||\boldsymbol{W}_i||_{max} \leq W\}$. *Let* $d = \max_i d_i$. *Then, there exists a FC ReLU binary network* $g(\boldsymbol{x})$ *of width* $\mathcal{O}(d \log |W|)$ *and depth* $\mathcal{O}(l \log |W|)$ *that can be pruned to* $\tilde{g}_f(\boldsymbol{x})$ *for every* $f \in \mathcal{F}_W$ *so that* $\tilde{g}_f(\boldsymbol{x}) = f(\boldsymbol{x})$ *for all* $\boldsymbol{x} \in \mathbb{R}^{d_0}$.

Proof. Note that there is no approximation error in any of the steps above. Therefore, we can just repeat the construction above for each of the l layers in $f(\boldsymbol{x})$. This results in a network $g(\boldsymbol{x})$ of width $\mathcal{O}(d \log |W|)$ and depth $\mathcal{O}(l \log |W|)$. A more precise description of the dimensions of each layer can be found in the statement of Theorem 2. □

We are now ready to prove Theorem 2 in its entirety by showing that our construction can be extended for networks with randomly initialized weights. We restate it here in a slightly simpler form for convenience.

Theorem 2. *Consider the set of FC ReLU networks with integer weights* $\mathcal{F}_W = \{f : f(\boldsymbol{x}) = \sigma(\boldsymbol{W}_l\sigma(\boldsymbol{W}_{l-1} \ldots \sigma(\boldsymbol{W}_1\boldsymbol{x}))),\ \boldsymbol{W}_i \in \mathbb{Z}^{d_i \times d_{i-1}},\ ||\boldsymbol{W}_i||_{max} \leq W\}$ *where* $W > 0$. *Let* $d = \max_i d_i$ *and let* $g(\boldsymbol{x})$ *be a randomly initialized binary network of width* $\Theta\left(d \log\left(\frac{(dl \log^2 |W|)}{\delta}\right)\right)$ *and depth* $\Theta(l \log |W|)$ *such that every weight is drawn uniformly from* $\{-1, +1\}$. *Then, for every* $f \in \mathcal{F}_W$, $g(\boldsymbol{x})$ *can be pruned to* $\tilde{g}_f(\boldsymbol{x})$ *so that* $\tilde{g}_f(\boldsymbol{x}) = f(\boldsymbol{x})$ *for all* $\boldsymbol{x} \in \mathbb{R}^{d_0}$ *with probability at least* $1 - \delta$.

Proof. Consider any one particular diamond structure in Figure 2. If we sample the weights of this network uniformly at random from $\{-1, +1\}$, then the probability that they are all 1 is at most $(1/2)^4$. If we make the network k times wider, the probability that such a diamond-gadget does not exist in the network is given by $(1-(1/2)^4)^k$. In other words, if A is the failure event, then $P(A) = \left(1-(1/2)^4\right)^k$. Note that there are $\Theta(dl\log^2|W|)$ such diamond structures in our network. By symmetry, the failure probability of each of them is identical. To have the overall probability of failure to be at most δ, taking the union bound we have that

$$dl\log^2|W| \cdot \left(1-(1/2)^4\right)^k \leq \delta$$

Hence, it suffices to have $k \geq \Theta\left(\log\left(\frac{(dl\log^2|W|)}{\delta}\right)\right)$. In other words, a randomly initialized binary network of width $\Theta\left(\log\left(\frac{(dl\log^2|W|)}{\delta}\right)\right)$ and depth $\Theta(l\log|W|)$ contains our deterministic construction with probability at least $1-\delta$. □

Remark 8. *Note that we need the randomness to just ensure the existence of a particular deterministic network - specifically $g(\boldsymbol{x})$ from Lemma 6. Therefore, a single random network is sufficient to ensure the guarantee for* every *$f \in \mathcal{F}_W$. This allows us to avoid using any union bound arguments.*

3.2.3 Putting everything together

First, note that by Lemma 2, to approximate any $f \in \mathcal{F}$ within $\varepsilon > 0$, it suffices to consider $\hat{g}(\boldsymbol{x})$ which is a finite-precision version of f where the precision of each weight is at most $p = \log(d^2l/\varepsilon)$. Now, applying the scaling trick in Equation (1) we can represent $\hat{g}(\boldsymbol{x})$ exactly as a scaled integer network *i.e.*,

$$\hat{g}(\boldsymbol{x}) = c^{-l}\sigma(c\widehat{\boldsymbol{W}}_l\sigma(c\widehat{\boldsymbol{W}}_{l-1}\ldots\sigma(c\widehat{\boldsymbol{W}}_1\boldsymbol{x})))$$

where $c = 10^p = (d^2l/\varepsilon)$ and all the weight matrices $c\widehat{\boldsymbol{W}}_i$ are integer. Since $\|\boldsymbol{W}_i\|_{\max} \leq 1$, it is clear that $\|c\widehat{\boldsymbol{W}}_i\|_{max} \leq c$. Therefore, applying Theorem 2 to the integer network $c^l\hat{g}(\boldsymbol{x})$ with $W = (d^2l/\varepsilon)$, we have the following. If $h(\boldsymbol{x}) = \sigma(\boldsymbol{B}_{l'}\sigma(\boldsymbol{B}_{l'-1}\ldots\sigma(\boldsymbol{B}_1\boldsymbol{x})))$ is a randomly initialized binary network of depth $\Theta(l\log(d^2l/\varepsilon))$ and width $\Theta\left(\log(d^2l/\varepsilon)\log\left(\frac{dl\log^2(d^2l/\varepsilon)}{\delta}\right)\right)$, then with probability at least $(1-\delta)$, it can be pruned to $\tilde{h}(\boldsymbol{x})$ so that $c^l\hat{g}(\boldsymbol{x}) = \tilde{h}(\boldsymbol{x})$ for any $\boldsymbol{x}$ in the unit sphere. Therefore, to approximate $\hat{g}(\boldsymbol{x})$ we simply push the scaling factor c^{-l} into the last layer $B_{l'}$ so that its weights are now scaled binary $\{\pm(\varepsilon/d^2l)^l\}$. Combining this with the approximation guarantee between $\hat{g}(\boldsymbol{x})$ and $f(\boldsymbol{x})$ completes the proof of Theorem 1.

3.3 Binary weights for classification

Theorem 2 can easily be extended for classification problems using finite-precision networks. Since $\text{sign}(\cdot)$ is positive scale-invariant, we no longer even require the final layer to be scaled. Applying the same argument as Sec 3.2.3 and then dropping the c^{-l} factor gives us the following corollary.

Corollary 1. *Consider the set of binary classification FC ReLU networks $\mathcal{F}$ of width d and depth l, where the weight matrices $\{\boldsymbol{W}_i\}_{i=1}^l$ are of finite-precision of at most p digits. Let $g(\boldsymbol{x}) = \text{sign}(\boldsymbol{B}_{l'}\sigma(\boldsymbol{B}_{l'-1}\ldots\sigma(\boldsymbol{B}_1\boldsymbol{x})))$ be a randomly initialized binary network with depth $l' = \Theta(lp)$ and width $d' = \Theta(dp\log(dlp/\delta))$ such that every weight is drawn uniformly from $\{-1,+1\}$. Then, with probability at least $1-\delta$, for every $f \in \mathcal{F}$, there exist pruning matrices $\{\boldsymbol{M}_i\}_{i=1}^{l'}$ such that $f(\boldsymbol{x}) = \tilde{g}(\boldsymbol{x})$ for any $\boldsymbol{x}$ where $\tilde{g}(\boldsymbol{x}) := \text{sign}((\boldsymbol{M}_{l'} \odot \boldsymbol{B}_{l'})\sigma((\boldsymbol{M}_{l'-1} \odot \boldsymbol{B}_{l'-1})\ldots\sigma((\boldsymbol{M}_1 \odot \boldsymbol{B}_1)\boldsymbol{x})))$.*

3.4 Observation on the power of zero

In the following, we try to understand if pruning is essential to the expressive power of binary networks. We will show that pruning binary networks is strictly more powerful than merely sign-flipping their weights. To see this, consider the set of all networks that can be derived by pruning 1-hidden layer binary networks of width m and input dimension d:

$$\mathcal{F}_{pruned}^m = \left\{ f : f(\boldsymbol{x}) = \sigma\left(\sum_{i=1}^d x_i\sigma\left(\sum_{j=1}^m v_j w_j^i\right)\right), \; w_j, v_j \in \{+1,-1,0\}\right\}.$$

Similarly, the set of all 1-hidden layer binary networks of the same width without pruning is given by

$$\mathcal{F}_{bin}^m = \left\{ f : f(\boldsymbol{x}) = \sigma\left(\sum_{i=1}^d x_i\sigma\left(\sum_{j=1}^m v_j w_j^i\right)\right), \; w_j, v_j \in \{+1,-1\}\right\}. \quad (3)$$

In the following proposition, we prove that $\mathcal{F}_{pruned}$ is a strictly richer class of functions indicating that pruning is an essential part of approximating classifiers.

Proposition 1. *The function $f(\boldsymbol{x}) = \sigma\left(\sum_{i=1}^d i \cdot x_i\right)$ satisfies $f(\boldsymbol{x}) \in \mathcal{F}_{pruned}^d$ and $f(\boldsymbol{x}) \notin \mathcal{F}_{bin}^d$, i.e., without pruning, $f(\boldsymbol{x})$ cannot be represented by a single layer binary network of width d.*

Proof. For simplicity, we consider the case when $x \geq 0$ so that the ReLU is equivalent to a linear activation. If

the functions are not equal for the non-negative orthant, then they are surely different. First, note that we recover $f(\boldsymbol{x})$ from $\mathcal{F}^d_{pruned}$ if we set $v_j = 1$ and $w^i_j = \mathbb{1}_{j \le i}$ for all i, j. Therefore, $f(\boldsymbol{x}) \in \mathcal{F}^d_{pruned}$ holds. To see that $f \notin \mathcal{F}^d_{bin}$, first note that we can replace $v_j w^i_j$ with $z^i_j \in \{+1, -1\}$ $\forall i, j$ in Equation (3). Hence, any $g \in \mathcal{F}^d_{bin}$ is of the form $g(\boldsymbol{x}) = \sigma\left(\sum_{i=1}^d x_i \sigma\left(\sum_{i=1}^d z^i_j\right)\right)$. Consider the coefficient of x_d in $f(\boldsymbol{x})$ which is $(d-1)$. Since z^d_j cannot be set to 0, the best approximation we can get using $g(\boldsymbol{x})$ is d or $d-2$. In fact, this holds for every odd term in $f(x)$. This completes the proof. □

Remark 9. *The above proposition suggests that pruning is more powerful than merely flipping signs of a binary network. In fact, the same argument can be extended for binary networks of any fixed width d and depth l to show that pruned networks are more expressive. However, it does not quantify this difference in expressivity.*

4 Conclusion

In this paper, we prove the Strong LTH for binary networks establishing that logarithmic overparameterization is sufficient for pruning algorithms to discover accurate subnetworks within random binary models. By doing this, we provide theory supporting the wide range of experimental work in the field, *e.g.*, scaled binary networks can achieve the best SOTA accuracy on benchmark image datasets (Diffenderfer and Kailkhura, 2021; Ramanujan et al., 2020). Moreover, we show that only the last layer needs to be scaled binary, while the rest of the network can be purely binary $\{\pm 1\}$. It is well known in the binary network literature that a *gain* term (scaling the weights) makes the optimization problem more tractable (Simons and Lee, 2019). While this is known empirically, it would be interesting to study this from a theoretical perspective so we can identify better algorithms to find binary networks of high accuracy.

References

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945.

Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., and Guttag, J. (2020). What is the state of neural network pruning? In *Proceedings of Machine Learning and Systems 2020*, pages 129–146.

Cheng, Y., Wang, D., Zhou, P., and Zhang, T. (2019). A Survey of Model Compression and Acceleration for Deep Neural Networks. *arXiv:1710.09282 [cs]*.

Courbariaux, M., Bengio, Y., and David, J.-P. (2015). Binaryconnect: Training deep neural networks with binary weights during propagations. *arXiv preprint arXiv:1511.00363*.

Deng, L., Li, G., Han, S., Shi, L., and Xie, Y. (2020). Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532.

Diffenderfer, J. and Kailkhura, B. (2021). Multi-prize lottery ticket hypothesis: Finding accurate binary neural networks by pruning a randomly weighted network. *arXiv preprint arXiv:2103.09377*.

Frankle, J. and Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. (2020). Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR.

Han, S., Mao, H., and Dally, W. J. (2015a). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Han, S., Pool, J., Tran, J., and Dally, W. J. (2015b). Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*.

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

Hassibi, B. and Stork, D. G. (1993). Second order derivatives for network pruning: Optimal Brain Surgeon. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *Advances in Neural Information Processing Systems 5*, pages 164–171. Morgan-Kaufmann.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., and Han, S. (2018). Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800.

He, Y., Zhang, X., and Sun, J. (2017). Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4114–4122.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2017). Quantized neural networks: Training neural networks with low precision weights

and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898.

Kidger, P. and Lyons, T. (2020). Universal approximation with deep narrow networks. In *Conference on Learning Theory*, pages 2306–2327. PMLR.

Klusowski, J. M. and Barron, A. R. (2018). Approximation by combinations of relu and squared relu ridge functions with ℓ^1 and ℓ^0 controls. *IEEE Transactions on Information Theory*, 64(12):7649–7656.

LeCun, Y., Denker, J. S., and Solla, S. A. (1990). Optimal Brain Damage. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 598–605. Morgan-Kaufmann.

Levin, A. U., Leen, T. K., and Moody, J. E. (1994). Fast Pruning Using Principal Components. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pages 35–42. Morgan-Kaufmann.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. (2016). Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*.

Malach, E., Yehudai, G., Shalev-Schwartz, S., and Shamir, O. (2020). Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pages 6682–6691. PMLR.

Mozer, M. C. and Smolensky, P. (1989). Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in neural information processing systems*, pages 107–115.

Orseau, L., Hutter, M., and Rivasplata, O. (2020). Logarithmic pruning is all you need. *Advances in Neural Information Processing Systems*, 33.

Pensia, A., Rajput, S., Nagle, A., Vishwakarma, H., and Papailiopoulos, D. (2020). Optimal lottery tickets via subsetsum: Logarithmic overparameterization is sufficient. *arXiv preprint arXiv:2006.07990*.

Perekrestenko, D., Grohs, P., Elbrächter, D., and Bölcskei, H. (2018). The universal approximation power of finite-width deep relu networks. *arXiv preprint arXiv:1806.01528*.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. (2020). What's Hidden in a Randomly Weighted Neural Network? *arXiv:1911.13299 [cs]*.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. (2016). Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer.

Scarselli, F. and Tsoi, A. C. (1998). Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. *Neural networks*, 11(1):15–37.

Simons, T. and Lee, D.-J. (2019). A review of binarized neural networks. *Electronics*, 8(6):661.

Stinchombe, M. (1989). Universal approximation using feed-forward networks with nonsigmoid hidden layer activation functions. *Proc. IJCNN, Washington, DC, 1989*, pages 161–166.

Wang, Y., Zhang, X., Xie, L., Zhou, J., Su, H., Zhang, B., and Hu, X. (2019). Pruning from Scratch. *arXiv:1909.12579 [cs]*.

Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. (2016). Learning structured sparsity in deep neural networks. *arXiv preprint arXiv:1608.03665*.

Wu, J., Leng, C., Wang, Y., Hu, Q., and Cheng, J. (2016). Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828.

Zhou, H., Lan, J., Liu, R., and Yosinski, J. (2019). Deconstructing lottery tickets: Zeros, signs, and the supermask. *arXiv preprint arXiv:1905.01067*.

Zhu, C., Han, S., Mao, H., and Dally, W. J. (2016). Trained ternary quantization. *arXiv preprint arXiv:1612.01064*.

Zhu, M. and Gupta, S. (2017). To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*.

A Appendix

A.1 Proof of Lemma 2

Before proving Lemma 2 in its entirety, we first extend Lemma 1 to approximate a neuron with $\log(d/\varepsilon)$-precision.

Lemma 7. *Consider a network $h(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T x)$ where $\boldsymbol{w} \in \mathbb{R}^d, ||\boldsymbol{w}|| \leq 1$ and $\varepsilon > 0$. Let $\hat{\boldsymbol{w}}$ be a coordinate-wise finite-precision truncation of $\boldsymbol{w}$ up to $\log(d/\varepsilon)$ digits and $\hat{g}(x) = \sigma(\hat{\boldsymbol{w}}x)$. Then we have that*

$$\max_{\boldsymbol{x} \in \mathbb{R}^d : ||\boldsymbol{x}|| \leq 1} ||h(\boldsymbol{x}) - \hat{g}(\boldsymbol{x})|| \leq \varepsilon.$$

Proof. Once again, by Lemma 1, we have that for each coordinate $|\boldsymbol{w}_i - \hat{\boldsymbol{w}}_i| \leq \varepsilon/d$. Therefore, $||\boldsymbol{w} - \hat{\boldsymbol{w}}|| \leq \sqrt{\sum_{i=1}^d \left(\frac{\varepsilon}{d}\right)^2} \leq \varepsilon$. Applying Cauchy-Schwarz with $||\boldsymbol{x}|| \leq 1$ completes the proof. □

We now extend the result to approximating a single layer with finite-precision.

Lemma 8. *Consider a network $h(\boldsymbol{x}) = \sigma(\mathbf{1}^T\sigma(\boldsymbol{W}_1\boldsymbol{x}))$ where $\boldsymbol{W}_1 \in \mathbb{R}^{d_1\times d_0}, ||\boldsymbol{W}_1|| \leq 1$ and $\varepsilon > 0$. Let $\widehat{\boldsymbol{W}_1}$ be a coordinate-wise finite-precision truncation of $\boldsymbol{W}_1$ up to $\log(d^2/\varepsilon)$ digits and $\hat{g}(x) = \sigma(\widehat{\boldsymbol{W}_1}x)$ where $d = \max\{d_0, d_1\}$. Then we have that*

$$\max_{\boldsymbol{x}\in\mathbb{R}^{d_0}:||\boldsymbol{x}||\leq 1} ||h(\boldsymbol{x}) - \hat{g}(\boldsymbol{x})|| \leq \varepsilon.$$

Proof. Note that for any $\boldsymbol{x} : ||\boldsymbol{x}|| \leq 1$,

$$\begin{aligned}
&\|h(\boldsymbol{x}) - \tilde{g}(\boldsymbol{x})\| \\
&= \|\sigma(\sum_{i=1}^{d_1}\sigma({W_1}^{(i)}x)) - \sigma(\sum_{i=1}^{d_1}\sigma({\widehat{W_1}}^{(i)}x))\| \\
&\leq \|\sum_{i=1}^{d_1}\sigma({W_1}^{(i)}x) - \sum_{i=1}^{d_1}\sigma({\widehat{W_1}}^{(i)}x)\| \\
&\qquad\qquad \text{(Since } \sigma \text{ is 1-Lipschitz)} \\
&\leq \sum_{i=1}^{d_1}\|\sigma({W_1}^{(i)}x) - \sigma({\widehat{W_1}}^{(i)}x)\| \\
&\leq \sum_{i=1}^{d_1}\frac{\varepsilon}{\max\{d_0,d_1\}} \\
&\leq \varepsilon
\end{aligned}$$

Since this holds for any $\boldsymbol{x}$, it also holds for the $\boldsymbol{x}$ that attains the maximum possible error which completes the proof. □

We are now ready to prove Lemma 2 which states that to approximate any FC ReLU network within ε, it suffices to simply consider weights with logarithmic precision.

Lemma 2. *Consider a network $h(\boldsymbol{x}) = \sigma(\boldsymbol{W}_l\sigma(\boldsymbol{W}_{l-1}\dots\sigma(\boldsymbol{W}_1\boldsymbol{x})))$ where $\boldsymbol{W}_i \in \mathbb{R}^{d_i\times d_{i-1}}$, $||\boldsymbol{W}_i|| \leq 1, d_i \leq d$ and $\varepsilon > 0$. Let $\hat{g}(\boldsymbol{x}) = \sigma(\widehat{\boldsymbol{W}}_l\sigma(\widehat{\boldsymbol{W}}_{l-1}\dots\sigma(\widehat{\boldsymbol{W}}_1\boldsymbol{x})))$ where $\widehat{\boldsymbol{W}}_i$ is a finite precision truncation of $\boldsymbol{W}_i$ up to $\log(d^2l/\varepsilon)$ digits. Then we have that*

$$\max_{\boldsymbol{x}\in\mathbb{R}^{d_0}:||\boldsymbol{x}||\leq 1} ||h(\boldsymbol{x}) - \hat{g}(\boldsymbol{x})|| \leq \varepsilon.$$

Proof. The proof follows inductively. First note that since operator norm is bounded by the frobenius norm, we have for the output of the first layer that,

$$\begin{aligned}
\|\sigma(\boldsymbol{W}_1\boldsymbol{x}) - \sigma(\widehat{\boldsymbol{W}}_1\boldsymbol{x})\| &\leq \|\boldsymbol{W}_1\boldsymbol{x} - \widehat{\boldsymbol{W}}_1\boldsymbol{x}\| \\
&\leq \|\boldsymbol{W}_1 - \widehat{\boldsymbol{W}}_1\| \\
&\leq \|\boldsymbol{W}_1 - \widehat{\boldsymbol{W}}_1\|_F \\
&\leq \varepsilon/l
\end{aligned}$$

Next, denote the output of the first layer by $\boldsymbol{x}_2 := \sigma(\boldsymbol{W}_1\boldsymbol{x})$, $\widehat{\boldsymbol{x}}_2 := \sigma(\widehat{\boldsymbol{W}}_1\boldsymbol{x})$. Repeating the calculation Note that the first term is bounded by ε/l since $||\boldsymbol{x}_2|| \leq 1$. Further, using the fact that $\|\widehat{\boldsymbol{W}}_2\| \leq 1$ and $\|\boldsymbol{x}_2 - \widehat{\boldsymbol{x}}_2\| \leq \varepsilon/l$ as proved above, we get that

$$\|\sigma(\boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x})) - \sigma(\widehat{\boldsymbol{W}}_2(\sigma(\widehat{\boldsymbol{W}}_1\boldsymbol{x}))\| \leq 2\varepsilon/l$$

By induction, for each layer $1 \leq i \leq l$ we have that for any $\boldsymbol{x} : \|\boldsymbol{x}\| \leq 1$,

$$\begin{aligned}
&\|\big(\sigma(\boldsymbol{W}_i\sigma(\boldsymbol{W}_{i-1}\dots\sigma(\boldsymbol{W}_1\boldsymbol{x})))\big) \\
&- \big(\sigma(\widehat{\boldsymbol{W}}_i\sigma(\widehat{\boldsymbol{W}}_{i-1}\dots\sigma(\widehat{\boldsymbol{W}}_1\boldsymbol{x})))\big)\| \leq \varepsilon\cdot(i/l)
\end{aligned}$$

Setting $i = l$ completes the proof. □