Neural Networks With Dense Weights Are Not Universal Approximators

Anonymous Authors[1]

Abstract

We investigate the approximation capabilities of dense neural networks. While universal approximation theorems establish that sufficiently large architectures can approximate arbitrary continuous functions when there are no restrictions on the weights, we show that dense neural networks do not possess this universality. Our argument is based on a model compression approach, combining the weak regularity lemma with an interpretation of feedforward networks as message passing graph neural networks. We consider ReLU neural networks subject to natural constraints on weights and input and output dimensions, which model a notion of dense connectivity. Within this setting, we demonstrate the existence of Lipschitz continuous functions that cannot be approximated by such networks. This highlights intrinsic limitations of neural networks with dense layers and motivates the use of sparse connectivity as a necessary ingredient for achieving true universality.

1. Introduction

Deep neural networks are central to modern machine learning, driving advances in vision, language, and scientific applications (LeCun et al., 2015). Recently, deep networks have grown increasingly large, enhancing their performance while increasing computational costs (Thompson et al., 2020), which motivates a focus on efficiency in both training and inference (Menghani, 2023). One of the most prominent approaches to improving efficiency is the use of sparse neural networks. Sparse models can match or surpass dense ones in accuracy, using less energy, memory, and training time (Mocanu et al., 2018; Evci et al., 2020; Hoefler et al., 2021; Peste, 2023). These properties have made sparsification methods such as pruning (LeCun et al., 1990; Medeiros and Barreto, 2007; Han et al., 2015; Lee et al., 2019) key in the use of deep networks on limited-resource devices and for lowering inference cost in large-scale systems (Cheng et al., 2018). Strikingly, even pruning 80–90% of parameters often leaves generalization (measured by test error) unaffected (Frankle and Carbin, 2019). Beyond efficiency, sparsity can enhance optimization at certain levels, act as a built-in regularizer, and reduce sensitivity to noisy data (He et al., 2022; Jin et al., 2022).

These empirical evidences raise the possibility that denseness itself may impose intrinsic limitations. While neural networks are known to approximate broad function classes (Yarotsky, 2017), it remains unclear whether such guarantees persist under dense connectivity constraints. This motivates the following central question: **Do dense deep neural networks have inferior expressivity?**

Our work focuses on the L^∞-approximation error for 1-Lipschitz continuous functions bounded by 1 on a d-dimensional domain. Within this setting, the approximation capabilities of neural networks have been extensively studied since the development of classical universal approximation theorems (Cybenko, 1989; Hornik, 1991). A central result is that any Lipschitz continuous (or more generally continuous) function on a d-dimensional compact domain can be approximated to arbitrary accuracy $\epsilon > 0$ by a sufficiently large ReLU network (Mhaskar, 1996; Pinkus, 1999). (Yarotsky, 2017) gave explicit error bounds for approximating functions in Sobolev spaces of given smoothness using deep and shallow ReLU networks. Notably, (Yarotsky, 2017) provides rigorous upper and lower bounds on the number of weights required for a given approximation error.

In contrast to prior work that studies the expressivity of general neural networks, we specifically investigate the expressivity limits of dense networks. We consider dense networks whose weight matrices are uniformly bounded, with both entrywise ℓ_∞ control and normalized ℓ_1 magnitude. These constraints enforce a strong form of denseness: nontrivial outputs require contributions from many neurons, so dense connectivity reflects genuine collective computation rather than a few dominating weights. By leveraging the graph structure of neural computation, we connect deep neural networks to the broader class of graph

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

neural networks (Gilmer et al., 2017b; Xu et al., 2019), for which a rich theoretical framework has been developed (Levie, 2023; Rauchwerger and Levie, 2025; Böker et al., 2023; Rauchwerger et al., 2025; Herbst and Jegelka, 2025; Finkelshtein et al., 2024; Kouchly et al., 2025) using dense graph limit theory (Lovász, 2012). Our analysis builds on the so-called *weak regularity lemma* (Frieze and Kannan, 1999; László and Szegedy, 2007) to obtain an implicit compression result, showing that – even as width increases (and depth is held fixed) – the expressive power of dense networks saturates at a fixed resolution, thereby revealing intrinsic barriers that scaling alone cannot overcome. Our theoretical framework is fundamentally different from both compression-based generalization analyses (Arora et al., 2018) and optimization-based perspectives such as in (Jacot et al., 2018). Instead, we study intrinsic expressivity constraints of dense architectures, independent of training, generalization, or optimization dynamics, by leveraging the precise characterizations provided by (Yarotsky, 2017) and structural graph-theoretic constraints inherent to dense architectures. We are not aware of any prior work that uses compression to derive expressivity limits, particularly for dense neural networks.

Contributions. We provide a theoretical account of a widely observed phenomenon: dense neural networks are fundamentally less expressive than their sparse counterpart. Our results can be summarized as follows.

1. Classical results achieve universality by allowing width to increase with the desired approximation accuracy. In contrast, we show that, under some conditions on the input dimension, dense networks of any fixed depth cannot exploit width alone to achieve universality; their expressive power saturates, revealing an inherent limitation of dense architectures (Theorem 5.2).
2. Our proof technique is not only novel but also offers a broadly applicable framework for analyzing neural networks. We identify two primary ways this technique can be adopted:
   - **General Approximation Method:** The method can be generalized to any subclass of networks that admits a compression guarantee – specifically, where there exists a small, bounded class that approximates the original subclass.
   - **Graph-Theoretic Perspective:** By treating the neural network as a computational graph, our approach utilizes tools from GNN theory – specifically the regularity lemma – to study architectural properties. This provides a blueprint for using GNN theory, and specifically regularity theorems, to analyze other aspects of neural networks.
3. As a byproduct, our analysis yields an implicit model compression scheme (Corollary 4.3) showing that every dense ReLU network can be approximated by a bounded-size network.

2. Preliminaries

Here, we provide graph-theoretic background, including kernels as a dense analogue of graphs (Lovász, 2012), introduce Euclidean and graph networks, propose a formal "strong" notion of dense networks, and recall definitions of network complexity and function approximation.

Graphs. For $n \in \mathbb{N}$, we write $[n] =: \{1, \dots, n\}$. An *attributed graph* (or *graph-signal*) is a pair $(G, \mathbf{f})$, where G is a weighted directed graph with node set $V = [n]$, edge set $E \subseteq V \times V$, and adjacency matrix $\mathbf{A} := (A_{i,j})_{i,j \in [n]}$, and $\mathbf{f} : V \to \mathbb{R}$ is a *node feature vector* (*signal*). The signal assigns to each node $v \in V$ an attribute $\mathbf{f}(v) \in \mathbb{R}$. Let $B > 0$, define a $[-B, B]$-weighted graph as a graph with edge weights in $[-B, B]$.

Partitions. Let μ denote the Lebesgue measure on $[0, 1]$. A *partition* of $[0, 1]$ is a sequence $\mathcal{P}_n = \{P_1, \dots, P_n\}$ of disjoint measurable sets such that $\bigcup_{j=1}^n P_j = [0, 1]$. The partition is an *equipartition* if $\mu(P_i) = \mu(P_j)$ for all $i, j \in [n]$. By $\mathcal{I}_n$, we denote the equipartition of $[0, 1]$ into n intervals. A partition $\mathcal{Q}_l$ is called a *refinement* of $\mathcal{P}_k$ if every $Q \in \mathcal{Q}_l$ is a subset of some $P \in \mathcal{P}_k$. The above definitions extend naturally to general intervals $[a, b]$ instead of $[0, 1]$.

Kernels. Kernels generalize the idea of $[-1, 1]$-weighted adjacency matrices to the continuous set of nodes $[0, 1]$. An *attributed kernel* (or *kernel-signal*) is a pair (K, f), where K is a *kernel*, i.e. a measurable function $K : [0, 1]^2 \to [-1, 1]$, and $f : [0, 1] \to \mathbb{R}$ is a measurable function called *feature vector* (or *signal*). By $\mathbb{1}_S$, we denote the indicator function of a set S. We call a kernel K a *step kernel* w.r.t. a partition $\mathcal{P}_n := \{P_i\}_{i=1}^n$ if $K(x, y) := \sum_{i,j \in [n]} c_{i,j} \mathbb{1}_{P_i \times P_j}(x, y)$ for some choice of coefficients $\{c_{i,j} \in [-1, 1]\}_{i,j \in [n]}$. Similarly, we call a signal f a *step signal* w.r.t. a partition $\mathcal{P}_n := \{P_i\}_{i=1}^n$ if $f(x) := \sum_{i \in [n]} c_i \mathbb{1}_{P_i}(x)$ for some choice of coefficients $\{c_i \in \mathbb{R}\}_{i \in [n]}$.

Graph-Induced Kernels. Any $[-B, B]$-weighted graph can be naturally associated with a kernel. Let G be a $[-B, B]$-weighted graph with adjacency matrix $\mathbf{A} = \{A_{i,j}\}_{i,j \in [n]}$ and signal $\mathbf{f}$. Consider the equipartition $\mathcal{I}_n = \{I_k\}_{k=1}^n$ of $[0, 1]$ into n intervals. The kernel K_G induced by G is defined by $K_G(x, y) = \sum_{i,j=1}^n \frac{A_{i,j}}{B} \mathbb{1}_{I_i}(x) \mathbb{1}_{I_j}(y)$, with the induced signal $f_{\mathbf{f}}(z) = \sum_{i=1}^n \mathbf{f}_i \mathbb{1}_{I_i}(z)$. This construction allows embedding the space of all graphs of all

sizes into the space of kernels,[1] where approximation properties are more naturally formulated.

Cut Norm. The cut norm, a fundamental distance measure in graph theory, was introduced by (Frieze and Kannan, 1999) and underpins the cut distance, which serves as the central notion of convergence in the theory of dense graph limits (Lovász, 2012). The *cut norm* of a measurable $K : [0,1]^2 \to \mathbb{R}$ and a measurable $\mathbf{f} : [0,1] \to \mathbb{R}$ are respectively defined as

$$\|K\|_\square = \sup_{U,S\subset[0,1]} \left| \int_{U\times S} K(x,y)dxdy \right|$$
$$\text{and} \quad \|f\|_\square := \sup_{S\subseteq[0,1]} \left| \int_S f(x)dx \right|.$$

A distance between graphs can be defined as the norm of the difference between their induced kernels.

Neural Networks. *Neural networks (NNs)*, also known as *multilayer perceptrons (MLPs)*, process vector-valued inputs in $\mathbb{R}^{d_0}$ using a collection of *learnable parameters* (or *weights*) $(\mathbf{W}, \mathbf{b})$. These consist of a sequence of weight matrices $\mathbf{W} = (W^{(\ell)})_{\ell=1}^L$, where $W^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$, and a sequence of bias vectors $\mathbf{b} = (b^{(\ell)})_{\ell=1}^L$ where $b^{(\ell)} \in \mathbb{R}^{d_\ell}$. We call $L \in \mathbb{N}$ the *depth* of the network and refer to each $0 \le \ell \le L$ as a *layer*. We call $d_\ell \in \mathbb{N}$ the *width* (or alternately the *dimension*) of ℓth layer and refer to each $i \in [d_\ell]$ as a *channel*. Specifically, we call d_0 and d_L the *input* and *output dimensions*, respectively, and refer to each $i \in [d_0]$ as an *input channel* and each $i \in [d_L]$ as an *output channel*. The tuple $(L, d_0, \ldots, d_L)$ is called the *network architecture*. The number of network parameters is $W := \sum_{\ell=1}^L d_\ell \cdot d_{\ell-1} + d_\ell$. The *network size* is $n := \frac{L+2}{L-1} \sum_{\ell=1}^{L-1} d_\ell$, serving as an effective size parameter; the rationale is given in Section 3.

Forward Propagation. We work throughout with a normalized parameterization in which each layer is scaled by a global factor depending on the network architecture. Given parameters $(\mathbf{W}, \mathbf{b})$, the associated depth-L ReLU network $\Theta_{(\mathbf{W},\mathbf{b})} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_L}$ is defined as follows. For an input $\mathbf{x} \in \mathbb{R}^{d_0}$, set $\mathbf{h}^{(0)} := \mathbf{x}$. The activation of the first hidden layer in channel $i \in [d_1]$ is

$$\mathbf{h}_i^{(1)} = \text{ReLU}\left(\frac{1}{d_0(L+2)} \sum_{j=1}^{d_0} \left(W_{ij}^{(1)} \mathbf{h}_j^{(0)} + b_i^{(1)} \right) \right). \quad (1)$$

[1]More accurately, the mapping from graphs to kernels is not one-to-one, and introduces equivalence classes of graphs that induce the same kernel. We will see that two equivalent computational graphs define the same function.

For layers $\ell = 2, \ldots, L-1$ and channel $i \in [d_\ell]$,

$$\mathbf{h}_i^{(\ell)} = \text{ReLU}\left(\frac{1}{n} \sum_{j=1}^{d_{\ell-1}} \left(W_{ij}^{(\ell)} \mathbf{h}_j^{(\ell-1)} + b_i^{(\ell)} \right) \right). \quad (2)$$

The output is computed as in Equation (2) with $\ell = L$, without ReLU. The network output is $\mathbf{h}^{(L)} = (\mathbf{h}_i^{(L)})_{i=1}^{d_L}$. The different normalization in the first layer reflects that d_0 and d_L are fixed in our asymptotic regime, while the effective network size n may grow arbitrarily.

B-Strongly Dense Networks. When weights are unconstrained, the normalized parameterization (in Equations (1) and (2)) does not limit generality, since any global scaling can be absorbed into the layer weights. However, under uniform weight bounds, it has structural consequences: it forces any nontrivial network—any network that does not map all inputs to near-zero outputs—to be *dense*: meaningful feature propagation requires many neurons to be active. We refer to this as a *strong* notion of denseness to distinguish it from weaker forms where a network might have many non-zero weights that are nonetheless large enough to allow a few neurons to dominate the output.

Definition 2.1 (B-Strongly Dense Network)**.** Let $B > 0$. A depth-L ReLU network $\Theta_{(\mathbf{W},\mathbf{b})}$ is called a *B-infinity normalized (B-IN) dense network*, or more concisely, a *B-strongly dense network*, if its parameters satisfy $|W_{i,j}^{(\ell)}| \le B$ and $|b_i^{(\ell)}| \le B$, for all layers $\ell \in [L]$, where B is a constant independent of the network size n.

In such networks, each neuron contributes $O(1/n)$ to the pre-activation, so no finite subset of neurons can dominate as $n \to \infty$. This contrasts with unbounded networks, where sparse representations with a few large weights remain possible. The parameters and the architecture of such networks are referred to as *dense parameters* and *dense architecture* (with weight bound B), respectively.

Graph and Kernel Neural Networks. *Message passing graph neural networks (MPNNs)* form a class of neural networks designed to process graph-structured data by iteratively updating node embeddings through the exchange of messages between nodes (Merkwirth and Lengauer, 2005; Gori et al., 2005; Xu et al., 2019; Gilmer et al., 2017a). (Levie, 2023; Rauchwerger and Levie, 2025) extend this framework canonically to operate on kernels. Since our construction is directly derived for kernels, we adopt their definitions. Unlike general trainable MPNNs, we consider the following special predefined network.

B-IR-MPNN. Let $B > 0$ and $L \in \mathbb{N}$. We define the B-amplified integral-ReLU MPNN (B-IR-MPNN) $\Phi_{B,L}$, with L layers, as the following mapping from attributed kernels

to node features. Given (K, f), define $\Phi_{B,L}(K, f) := f^{(L)}$, where with $f^{(0)}(x) := f(x)$ and for $\ell \in [L-1]$:

$$f^{(\ell)}(x) := B \cdot \mathrm{ReLU}\left(\int_{[0,1]} K(x,y) f^{(\ell-1)}(y) dy\right). \quad (3)$$

The output is computed as in Equation (3) with $\ell = L$, without ReLU. This definition is the canonical extension of MPNNs from graphs to kernels. For the definition of MPNNs on graphs and the equivalency to MPNNs on kernels see Appendix E.

Functions and Approximation. Our work focuses on the limitations of approximating Lipschitz continuous functions $[0,1]^d \to \mathbb{R}$ via B-strongly dense ReLU networks. Let $\mathrm{C}(d, d')$ denote the space of all continuous functions mapping $[0,1]^d$ to $\mathbb{R}^{d'}$. The *approximation error* between two functions f and g is $\|f - g\|_\infty = \operatorname{ess\,sup}_{\mathbf{x}\in[0,1]^d} \|f(\mathbf{x}) - g(\mathbf{x})\|_\infty$. Here, the outer $\|\cdot\|_\infty$ denotes the supremum over the input domain $[0,1]^d$, while the inner $\|\cdot\|_\infty$ denotes the standard ℓ_∞ norm on the output space (i.e., the maximum absolute value over coordinates). A function $f : [0,1]^d \to \mathbb{R}^{d'}$ is called *L-Lipschitz continuous* if there exists a constant $L \geq 0$ such that $\|f(\mathbf{x}) - f(\mathbf{y})\|_\infty \leq L \|\mathbf{x} - \mathbf{y}\|_\infty$ for all $\mathbf{x}, \mathbf{y} \in [0,1]^d$. Denote the space of 1-Lipschitz continuous functions bounded by 1 as $\mathrm{Lip}(d, d')$.

Universal Approximation. A standard way to define the expressive power of a class of neural networks is via universal approximation.

Definition 2.2. Let $\mathcal{NN}$ and $\mathcal{F}$ be two sets of measurable functions $\Omega \to \mathbb{R}^d$ defined on the measure space Ω. We say that $\mathcal{NN}$ is a *universal approximator* of $\mathcal{F}$ if for every $\epsilon > 0$ and $f \in \mathcal{F}$ there exists $\Theta \in \mathcal{NN}$ such that

$$\|f - \Theta\|_\infty := \operatorname*{ess\,sup}_{x\in\Omega} \|f(x) - \Theta(x)\|_\infty < \epsilon.$$

A classical result states that the class of neural networks with arbitrary width and predefined fixed depth is a universal approximator of $\mathrm{C}(d, d')$ (Leshno et al., 1993; Mhaskar, 1996; Pinkus, 1999). In our work, we take $\Omega = [0,1]^{d_0}$, choose $\mathcal{NN}$ to be a class of strongly dense neural networks (to be defined in Section 3) and show that it is not a universal approximator of $\mathcal{F} = \mathrm{Lip}(d_0, d_L)$.

Network Complexity and the VC Dimension. A classical way to formalize the complexity of a hypothesis class of neural networks is through its *VC-dimension*. The VC-dimension, which quantifies the largest set of points in an input space $\mathcal{X}$ that a hypothesis class $\mathcal{H}$ (a class of functions from $\mathcal{X}$ to $\{0,1\}$) can shatter, i.e., realize all possible Boolean labelings. If $|\{(h(x_1), \ldots, h(x_m)) : h \in \mathcal{H}\}| = 2^m$, we say $\mathcal{H}$ shatters the set $\{x_1, \ldots, x_m\}$. The Vapnik–Chervonenkis dimension of $\mathcal{H}$, denoted $\mathrm{VCdim}(\mathcal{H})$, is the size of the largest shattered set or ∞, if there is no such maximal set.

Note that our complexity measures are largely insensitive to the specific choice of piecewise linear activation function. In fact, (Yarotsky, 2017), Proposition 1, shows that any such activation with finitely many breakpoints can be simulated by ReLU networks with only a constant-factor increase in the number of units and weights. Thus, focusing on ReLU does not entail any real loss of generality.

3. Deep Neural Networks as MPNNs

Deep networks are often described algebraically – through layers, weights, and nonlinearities. Yet there is another perspective: every computation carried out by a network implicitly constructs a graph. Each node represents a feature channel and its value an activation, while each edge encodes a single weight of a weight matrix. Any feedforward network can thus be implemented via message passing, which propagates information through a specially designed directed graph – its *computational graph*.

In our analysis, we compare "large" neural networks to "small" neural networks that compress them. To define a distance between the computational graphs of networks of different sizes, we embed graphs of all sizes into the space of kernels, where there is linear structure and a natural norm. Rather than explicitly defining a computational graph and then inducing a corresponding kernel, we directly specify the class of kernels that encode valid neural network computations. These kernels, which we call *computational kernels*, are constructed so that, for each dense feedforward neural network, message passing with ReLU nonlinearities on the corresponding kernel exactly reproduces the forward propagation of the network. For completeness, we provide the equivalent graph-based definition in Appendix E. All results in the current section are proved in Appendix A.

Fixed-Width Networks. To simplify the presentation and subsequent analysis, we restrict the analysis to networks with fixed hidden width, i.e., $d_\ell = d$ for all $\ell = 1, \ldots, L-1$. In this case, the network size is $n = (L+2)d$, the number of parameters is $(L-1)d^2 + (d_0 + d_L + L - 1)d + d_L$, and the forward propagation rule becomes, for all $\ell \in [L]$ and $i \in [d_\ell]$,

$$\mathbf{h}_i^{(\ell)} = \mathrm{ReLU}\left(\frac{1}{d_{\ell-1}(L+2)} \sum_{j=1}^{d_{\ell-1}} \left(W_{ij}^{(\ell)} \mathbf{h}_j^{(\ell-1)} + b_i^{(\ell)}\right)\right).$$

Note that we can absorb the $(L+2)$ factor by choosing $B = (L+2)C$ for some $C > 0$. For simplicity, we assume that d is divisible by both d_0 and d_L; this assumption is not essential and can be removed at the cost of additional notation. The space of all such networks, with any value of

d, is denoted by $\mathcal{NN}(L, d_0, d_L)$ and similarly, the space of all such dense networks with weights and biases bounded in $[-B, B]$ by $\mathcal{NN}_{\text{dense}}(B, L, d_0, d_L)$.

The Computational Kernel of a Neural Network. We call a kernel $K : [0,1]^2 \to [-1,1]$ a depth-L *computational kernel* with respect to the parameters $L, d_0, d_L \in \mathbb{N}$ and $B \in \mathbb{R}$ with $B \geq L+2$ if K satisfies the following four conditions (for geometric interpretation, consult Figure 1). Each condition encodes a specific structural property of fixed-width feedforward networks, including layerwise connectivity, normalization of weights, and the representation of bias terms.

- *Condition 1.* There exists $n \in \mathbb{N}$ divisible by $M(L+2)$, where M is the least common multiple of d_0 and d_L, such that K is a step kernel with respect to the interval equipartition $\mathcal{I}_n$ of $[0,1]$ into n intervals.

  Notation: $\mathcal{I}_n$ is a refinement of the interval equipartition $\mathcal{U}_{L+2} := \{U^{(0)}, \ldots, U^{(L)}, U^{(\text{bias})}\}$ of $[0,1]$ into $L+2$ intervals of length $1/(L+2)$, where each $U^{(\ell)}$, $\ell = 0, \ldots, L$, is called a *layer*, and $U^{(\text{bias})}$ is called a *bias*. We call $\mathcal{U}_{L+2}$ the *layer partition*. For each ℓ, define $\mathcal{R}_d^{(\ell)} := \{I \in \mathcal{I}_n \mid I \subseteq U^{(\ell)}\}$. Here, $\mathcal{R}_d^{(\ell)}$ consists of $d := n/(L+2)$ intervals. The partition $\mathcal{R}_d^{(0)}$ is the refinement of the coarser interval equipartition $\mathcal{C}_{d_0}^{\text{in}}$ of the interval $U^{(0)}$ into d_0 intervals of length $1/(L+2)d_0$ called the *input partition*. The partition $\mathcal{R}_d^{(L)}$ is a refinement of the coarser interval equipartition $\mathcal{C}_{d_L}^{\text{out}}$ of the interval $U^{(L)}$ into d_L intervals of length $1/(L+2)d_L$ called the *output partition*.

- *Condition 2.* The kernel satisfies $K(x,y) = K(x,y')$ whenever y and y' belong to the same interval of $\mathcal{C}_{d_0}^{\text{in}}$. Moreover, $K(x,y) = K(x',y)$ whenever x and x' belong to the same interval of $\mathcal{C}_{d_L}^{\text{out}}$. In addition, $K(x,y) = K(x,y')$ whenever y and y' belong to $U^{(\text{bias})}$.

- *Condition 3.* The kernel satisfies $K(x,y) = 0$ whenever either $y \in U^{(\ell)}$ and $x \notin U^{(\ell+1)}$ for some $\ell \in \{0, \ldots, L-1\}$, or $y \in U^{(\text{bias})}$ and $x \in U^{(0)}$, or $x \in U^{(\text{bias})}$ and $y \notin U^{(\text{bias})}$.

- *Condition 4.* For any $x, y \in U^{(\text{bias})}$, $K(x,y) = (L+2)/B$.

We call d_0 the *input dimension*, d_L the *output dimension*, and refer to $d = n/(L+2)$ as the *hidden dimension* for $\ell = 1, \ldots, L-1$. We call n the *size* of the computational kernel. By $\mathcal{CK}(B, L, d_0, d_L)$, we denote the collection of all depth-L computational kernels as defined above.

Remark 3.1. Note that for any computational kernel there is a dense neural network that induces it. Similarly, every computational input signal is induced by some input to the network.

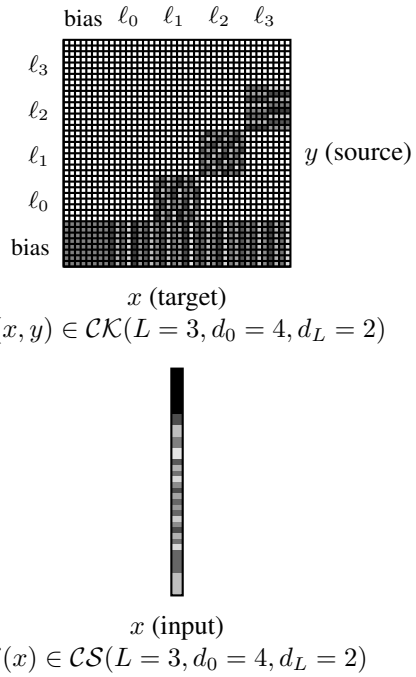

Figure 1. A computational kernel-signal with $d = 8$. The kernel K is nonzero only on blocks linking consecutive layers and bias-to-layer blocks; visible stripes in the first/last layer blocks reflect constancy over input/output coarse intervals. The signal f shares the same coarse-interval structure.

Computational Signals. While the kernel encodes weights, the signal encodes activations – inputs and propagated values. Given parameters L, d_0, d_L, a *computational signal* is any step signal with respect to the partition $\cup_{\ell=1}^{L-1} \mathcal{R}_d^{(\ell)} \cup \mathcal{C}_{d_0}^{\text{in}} \cup \mathcal{C}_{d_L}^{\text{out}} \cup \{U^{(\text{bias})}\}$. Denote the space of computational signals by $\mathcal{CS}(L, d_0, d_L)$. We define a *computational input signal* as any computational signal satisfying $f(x) = 1$ for all $x \in U^{(\text{bias})}$ and $f(x) = 0$ for all $x \notin U^{(0)} \cup U^{(\text{bias})}$. The values of f on the intervals of $\mathcal{C}_{d_0}^{\text{in}}$ are interpreted as the input vector to the network, while the constant value on $U^{(\text{bias})}$ represents a bias signal that is present at every layer. *Condition 4* in the definition of a computational kernel, i.e., $K(x,y) = (L+2)/B$ for all $x, y \in U^{(\text{bias})}$, ensures that this bias signal is unchanged during propagation, i.e., under successive applications of the computational kernel on the signal.

We call a pair in $\mathcal{CK}(B, L, d_0, d_L) \times \mathcal{CS}(L, d_0, d_L)$ a *computational kernel-signal*.

Computational Kernel Induced by Network Parameters. Given a dense neural network, one can induce a computational kernel on which the B-IR-MPNN imple-

ments the forward propagation of the neural network. Let $(\mathbf{W}, \mathbf{b})$ be the dense parameters of the neural network $\Theta_{(\mathbf{W},\mathbf{b})} \in \mathcal{NN}_{\text{dense}}(B, L, d_0, d_L)$, with hidden dimension d divisible by d_0 and d_L. Denote $n = d(L+2)$. We define the computational kernel $K = K_{(\mathbf{W},\mathbf{b})} \in \mathcal{CK}(B, L, d_0, d_L)$ *induced* by $\Theta_{(\mathbf{W},\mathbf{b})}$ as follows. For every $(x, y) \in [0, 1]^2$:

- $K(x, y) = W^{(1)}_{i,j}/B$ if y is in the jth interval of $\mathcal{C}^{\text{in}}_{d_0}$ and x is in the ith interval of $\mathcal{R}^{(1)}_d$.
- $K(x, y) = W^{(L)}_{i,j}/B$ if y is in the jth interval of $\mathcal{R}^{(L-1)}_d$ and x is in the ith interval of $\mathcal{C}^{\text{out}}_{d_L}$.
- For $\ell = 2, \ldots, L-1$, $K(x, y) = W^{(\ell)}_{i,j}/B$ if y is in the jth interval of $\mathcal{R}^{(\ell-1)}_d$ and x is in the ith interval of $\mathcal{R}^{(\ell)}_d$.
- For $\ell = 2, \ldots, L-1$, $K(x, y) = b^{(\ell)}_i/B$ if y is in $U^{(\text{bias})}$ and x is in the ith interval of $\mathcal{R}^{(\ell)}_d$. Moreover, $K(x, y) = b^{(L)}_i/B$ if y is in $U^{(\text{bias})}$ and x is in the ith interval of $\mathcal{C}^{\text{out}}_{d_L}$.
- For any $x, y \in U^{(\text{bias})}$, $K(x, y) = (L+2)/B$.

Given an input vector $\mathbf{x} = (x_1, \ldots, x_{d_0}) \in \mathbb{R}^{d_0}$ to the neural network, we define the *induced input signal* as the computational input signal $f_{\mathbf{x}}$ satisfying $f_{\mathbf{x}}(v) = x_j$ whenever v is in the jth interval of $\mathcal{C}^{\text{in}}_{d_0}$. In the above construction, when writing "the jth interval of the partition..." we implicitly assume that the intervals are always sorted in increasing order.

Networks as MPNNs. We can now state the key equivalence: Lemma 3.2 shows that the output of a B-strongly dense neural network can be represented as the output of an L-layer B-IR-MPNN applied to its induced computational kernel. This allows us to import tools from graph theory in Section 4.

Lemma 3.2. *Let $(\mathbf{W}, \mathbf{b})$ be the dense parameters of $\Theta_{(\mathbf{W},\mathbf{b})} \in \mathcal{NN}_{\text{dense}}(B, L, d_0, d_L)$. Then, for any input $\mathbf{x} \in \mathbb{R}^{d_0}$ and output channel $i \in [d_L]$, we have*

$$\Theta_{(\mathbf{W},\mathbf{b})}(\mathbf{x})_i = \Phi_{B,L}\big(K_{(\mathbf{W},\mathbf{b})}, f_{\mathbf{x}}\big)(v),$$

whenever v is in the ith interval of $\mathcal{C}^{\text{out}}_{d_L}$.

4. Model Compression

Modern deep learning models often contain millions or even billions of parameters, making them computationally expensive and memory-intensive. *Model compression* – a family of techniques including pruning (LeCun et al., 1990), quantization (Jacob et al., 2018), and distillation (Hinton et al., 2015) – has shown experimentally that smaller networks can frequently match the performance of much larger ones. Our approach to compression differs fundamentally from these existing methods in objective. While these methods propose empirical heuristics to compress specific trained models, our goal is to provide a deterministic guarantee that *any* network in a given class, regardless of whether it was trained or how it was initialized, can be approximated by a bounded-size network. This guarantee will be used in Section 5 to establish a formal barrier to the universal approximation power of dense neural networks. The full proofs of results from this section are presented in Appendix B.

Computational Cut Distance. In order to define compression, we first need to define a measure of similarity between the compressed and original object. In our analysis, two computational kernels are considered equivalent if one can be obtained from the other using a "permutation of the vertices" that preserves the layer structure. Next, we formalize this idea more accurately. Let $\mathcal{S}_L$ denote the set of measure-preserving bijections $\phi : [0, 1] \to [0, 1]$ such that each $U^{(\ell)}$, $\ell = 1, \ldots, L-1$, is invariant under ϕ, and $\phi(x) = x$ for every $x \in U^{(0)} \cup U^{(L)} \cup U^{(\text{bias})}$.[2] Define $J^{\phi}(x, y) := J(\phi(x), \phi(y))$. The *computational cut distance* between the computational kernel-signals $(K, f), (J, g)$ is defined to be

$$\delta^{\text{comp}}_{\square}((K, f), (J, g)) := \inf_{\phi \in \mathcal{S}_L} (\|K - J^{\phi}\|_{\square} + \|f - g \circ \phi\|_{\square}).$$

We define similarly $\delta^{\text{comp}}_{\square}(K, J) := \inf_{\phi \in \mathcal{S}_L} \|K - J^{\phi}\|_{\square}$. The computational cut distance is a pseudometric. MPNNs were shown in (Levie, 2023) to be Lipschitz continuous with respect to the cut norm, with a further extension in (Rauchwerger and Levie, 2025). In this work, we adapt the techniques of (Levie, 2023; Rauchwerger and Levie, 2025) to our setting in order to obtain an explicit bound on the Lipschitz constant of B-strongly dense ReLU networks w.r.t. the computational cut distance.

Theorem 4.1 (Computational Cut Distance Lipschitz Continuity)**.** *Let $B > 0$ and let $\Phi_{B,L}$ be the L-layer B-IR-MPNN. Let $(K, f), (J, g) \in \mathcal{CK}(B, L, d_0, d_L) \times \mathcal{CS}(L, d_0, d_L)$ be two computational kernel-signals. Let*

$$\text{Out}_{K,f} := \mathbb{1}_{U^{(L)}} \Phi_{B,L}(K, f),$$
$$\text{Out}_{J,g} := \mathbb{1}_{U^{(L)}} \Phi_{B,L}(J, g).$$

Then

$$\|\text{Out}_{K,f} - \text{Out}_{J,g}\|_{\square} \leq 2^L B^L \, \delta^{\text{comp}}_{\square}\big((K, f), (J, g)\big).$$

Note that $\|\text{Out}_{K,f} - \text{Out}_{J,g}\|_{\square}$ is a signal cut norm, as $(\text{Out}_{K,f} - \text{Out}_{J,g}) : [0, 1] \to \mathbb{R}$.

[2] More accurately, ϕ is a measure preserving bijection between two co-null sets of $[0, 1]$, as our analysis is always up to sets of measure zero. Measure preserving means that $\mu(A) = \mu(\phi(A))$ for measurable subsets A of the domain of ϕ.

Computational Kernel Weak Regularity Lemma. Theorem 4.2 establishes that any computational kernel, regardless of its original size or complexity, can be approximated by a smaller, structured kernel. Specifically, we utilize a variant of the weak regularity lemma (Frieze and Kannan, 1999) for kernels (László and Szegedy, 2007) to approximate any kernel by a step kernel whose complexity – measured by the number of steps in its partition – depends solely on the desired error ϵ in the computational cut distance. Building on techniques from (Levie, 2023; Rauchwerger and Levie, 2025), we refine this step kernel to preserve the specific neural bipartite layer structure, as well as the fixed input and output dimensions.

Theorem 4.2. *Let $K \in \mathcal{CK}(B, L, d_0, d_L)$ be an L-layer computational kernel. For any $\epsilon > 0$, there exists an L-layer computational kernel $K' \in \mathcal{CK}(B, L, d_0, d_L)$ with hidden dimension*

$$d = 8ML \left\lceil \frac{2^{2\lceil 16/\epsilon^2 \rceil}}{\epsilon} \right\rceil,$$

where M is the least common multiple of d_0 and d_L, such that $\delta_{\square}^{\text{comp}}(K, K') < \epsilon$.

This structural reduction allows us to represent the information of an arbitrarily wide network using a much smaller hidden dimension. When combined with the stability analysis of the forward pass presented in Theorem 4.1 – showing that B-IR-MPNNs are Lipschitz continuous with respect to the computational cut distance – this leads directly to our main compression result. Corollary 4.3 provides a uniform bound on the size of the approximating network that is independent of the original network's width.

Corollary 4.3 (Network Architecture Approximation)**.** *Let $L, d_0, d_L \in \mathbb{N}$ with $L \geq 2$, let $B \geq L + 2$, and $\epsilon > 0$. Then, for any network $\Theta \in \mathcal{NN}_{\text{dense}}(B, L, d_0, d_L)$ there is a network $\Theta' \in \mathcal{NN}_{\text{dense}}(B, L, d_0, d_L)$ with hidden dimension*

$$d = 8ML \left\lceil \frac{1}{\epsilon} \cdot 2^{2\left\lceil \frac{16((L+2)d_L)^2(2B)^{2L}}{\epsilon^2} \right\rceil} (L+2) d_L (2B)^L \right\rceil,$$

such that $\|\Theta - \Theta'\|_\infty < \epsilon$.

Why Not Monte Carlo Compression? A natural alternative to our approach would be to approximate dense networks by randomly subsampling hidden layer neurons. This approach is often called *randomized numerical linear algebra*. However, Monte Carlo sampling does not yield uniform guarantees: the approximation is only successful in high probability, and the event of success depends on the input to the network. Hence, different inputs would require different events of subsampled networks, and the intersection of all events in general can be the empty set, preventing simultaneous control of the approximation error on all inputs. In contrast, the weak regularity lemma provides deterministic and input-independent compression guarantees.

5. Implications for Approximation

The compression phenomenon established in Section 4 reveals that any dense ReLU network – no matter how large – can be approximated by a bounded-size network in the computational cut distance. This finding hints at a fundamental limitation: if expressive power saturates at a finite level of architectural complexity, then sufficiently large networks cannot continue to gain representational capacity merely by adding width. In this section, we make this intuition precise by identifying explicit conditions under which dense deep ReLU networks *fail to be universal approximators*.

VC-Dimension Based Lower Bounds on Expressivity. The first step is to relate functional approximation capacity to combinatorial complexity. Recall that for a hypothesis class $\mathcal{H}$ of Boolean functions on $[0,1]^d$, the *VC-dimension* is the largest number of points that $\mathcal{H}$ can shatter. When $\mathcal{H}$ consists of thresholded outputs of a ReLU network of fixed architecture but variable weights, Theorem 8.7 in (Anthony and Bartlett, 2009) provides the bound $\text{VCdim}(\mathcal{H}) \leq c \cdot W^2$, where W is the total number of weights and $c > 0$ is a universal constant independent of the architecture. This implies that representational capacity grows at most quadratically in the number of parameters. Theorem 7 of (Harvey et al., 2017) presents an explicit bound that improves upon the classical VC-dimension bound, showing that the universal constant c is in fact small.

The next theorem provides a quantitative lower bound on the number of computational units necessary for universal approximation, building on Theorem 4(a) in (Yarotsky, 2017). Our contribution lies in adapting this bound to our setting, and expressing it in terms of a single implicit constant. Recall that $\text{Lip}(d_0, d_L)$ is the space of 1-Lipschitz continuous functions $[0,1]^{d_0} \to \mathbb{R}^{d_L}$ bounded by 1.

Theorem 5.1. *For any $\epsilon \in (0, 1/3)$, a ReLU network architecture capable of approximating every function $f \in \text{Lip}(d_0, d_L)$ with error bounded by ϵ must have at least $c^{-1/2}(6\epsilon)^{-d_0/2}$ parameters, where c is the absolute constant defined above.*

This result provides a fundamental trade-off: as the input dimension d_0 grows, the number of parameters required for universality increases exponentially.

Approximation Limits of Dense Neural Networks. We now connect the above expressivity lower bound with the compression result. Theorem 5.1 gives a lower bound on the number of parameters required for an ϵ approximation, while Corollary 4.3 shows that neural networks that achieve

ϵ approximation effectively have a uniformly bounded number of parameters. As it turns out, in certain settings, the lower bound in Theorem 5.1 is higher than the upper bound entailed by Corollary 4.3. This shows that dense neural networks are not universal approximators of $\mathrm{Lip}(d_0, d_L)$.

Theorem 5.2 (Expressivity Bound of Strongly Dense Neural Networks)**.** *Let $L, d_L \in \mathbb{N}$ with $L \geq 2$, let $B \geq L+2$, and let c be the constant from Theorem 5.1. Let d_0 satisfy*

$$d_0 \geq 17 \log_2\big(c^{1/2} L^3 (L+2)^2 d_L^4 (2B)^{2L}\big) + 17 \cdot 2^{14} (L+2)^2 d_L^2 (2B)^{2L} + 306.$$

Then $\mathcal{NN}_{\mathrm{dense}}(B, L, d_0, d_L)$ is not a universal approximator of $\mathrm{Lip}(d_0, d_L)$.

We interpret Theorem 5.2 as follows. Since general neural networks of a fixed depth are universal approximators of $\mathrm{C}(d_0, d_L)$, they are also universal approximators of $\mathrm{Lip}(d_0, d_L)$. One might expect that neural networks with linear layers bounded by some universal constant $C > 1$ in their induced infinity norm, i.e. $\|W^{(\ell)}\|_\infty < C$, are also universal approximators of $\mathrm{Lip}(d_0, d_L)$.[3] In our definition of strongly dense neural networks, if one chooses $B = (L+2)C$ then the infinity norms of the linear layers are bounded by C. Hence, one might naively anticipate that $\mathcal{NN}_{\mathrm{dense}}(B, L, d_0, d_L)$ is a universal approximator of $\mathrm{Lip}(d_0, d_L)$. However, Theorem 5.2 shows that this is false, given that the input dimension is large enough (but still independent of n).

Connection to Double Descent and Modern Deep Learning. Our results in Theorem 5.2 offer, partially, a theoretical lens through which to view the double descent phenomenon. Classical bias-variance logic suggests that as the parameter count surpasses the sample size, the model's variance should diverge, causing the model to overfit the data and impair the generalization capacity. However, we show that for strongly dense networks, the realizable capacity actually plateaus, which supports the empirical observation that there is no unbounded growth of variance. We note that in practical deep learning, the density of the neural network is often implicitly enforced through techniques such as weight decay or layer normalization.

Experiments. We empirically validate our results on MNIST using one-hidden-layer ReLU networks trained with Adam; full details are in Appendix D. Standard networks closely approximate the target function, with training accuracy stabilizing around 98–99%. Dense networks – where weights are clamped to $[-10/d_{i-1}, 10/d_{i-1}]$ after each update – fail to do so, plateauing at 90–91% across all widths (Figure 2), supporting that density constraints create a hard barrier to approximation.

[3] In fact, as far as we know, the following is an open question: There exists a constant $C > 1$, such that the space of neural networks with linear layers $W^{(\ell)}$ having induced infinity norm $\|W^{(\ell)}\|_\infty < C$ is a universal approximator of $\mathrm{Lip}(d_0, d_L)$. We note that networks with sort activations are known to satisfy this property (Anil et al., 2018; Tanielian et al., 2020).

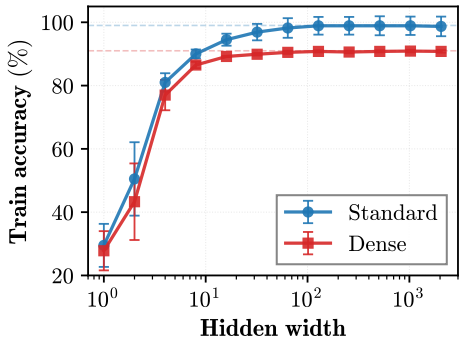

Figure 2. Train accuracy on MNIST.

6. Conclusion

We presented a framework linking graph-regularity-based compression of dense ReLU networks with VC-dimension lower bounds on expressivity. The combination yields explicit conditions under which depth- and weight-bounded dense networks cease to be universal, revealing a saturation effect: beyond a certain scale, width no longer increases representational capacity. These findings suggest that expressivity is often governed more by architectural structure than by size alone. While our present results apply to fully connected feedforward architectures, extending our compression-based analysis to structured families – such as convolutional or attention-based networks – offers a promising direction for developing a unified theory for expressivity saturation in deep learning.

7. Ethics Statement

This paper is theoretical in nature and does not involve datasets containing human or animal subjects. Consequently, it does not raise any direct ethical concerns.

8. Reproducibility Statement

All proofs of the theorems stated in this work are included in the appendix. Upon acceptance, we will release the code together with comprehensive documentation to reproduce every experiment described in the paper. This ensures that the results presented in both the main text and the appendix are fully replicable.

9. Large Language Models Discloser

LLMs were not used as part of the core methodology. Any use was limited to writing assistance and does not affect

scientific content or originality.

References

C. Anil, J. Lucas, and R. B. Grosse. Sorting out lipschitz function approximation. In *Int. Conference on Machine Learning (ICML)*, 2018.

M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, USA, 1st edition, 2009. ISBN 052111862X.

S. Arora, R. Ge, B. Neyshabur, and Y. Zhang. Stronger generalization bounds for deep nets via a compression approach, 2018.

J. Böker, R. Levie, N. Huang, S. Villar, and C. Morris. Fine-grained expressivity of graph neural networks. *Advances in Neural Information Processing Systems (NIPS)*, 2023.

Y. Cheng, D. Wang, P. Zhou, and T. Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018.

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

U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen. Rigging the lottery: Making all tickets winners. In H. D. III and A. Singh, editors, *Int. Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 2943–2952. PMLR, 13–18 Jul 2020.

B. Finkelshtein, u. u. Ceylan, M. Bronstein, and R. Levie. Learning on large graphs using intersecting communities. In *Advances in Neural Information Processing Systems (NIPS)*, 2024.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Int. Conf. on Learning Representations (ICLR)*, 2019.

A. Frieze and R. Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, Feb. 1999.

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Int. Conference on Machine Learning (ICML)*, pages 1263–1272, 2017a.

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. *Int. Conference on Machine Learning (ICML)*, pages 1263–1272, 2017b.

M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *IJCNN*, 2005.

S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems (NIPS)*, 28, 2015.

N. Harvey, C. Liaw, and A. Mehrabian. Nearly-tight VC-dimension bounds for piecewise linear neural networks. In S. Kale and O. Shamir, editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 1064–1068. PMLR, 07–10 Jul 2017.

Z. He, Z. Xie, Q. Zhu, and Z. Qin. Sparse double descent: Where network pruning aggravates overfitting. In *Int. Conference on Machine Learning (ICML)*, 06 2022. doi: 10.48550/arXiv.2206.08684.

D. Herbst and S. Jegelka. Higher-Order Graphon Neural Networks: Approximation and Cut Distance. *Int. Conf. on Learning Representations (ICLR)*, 2025. Spotlight Paper. arXiv:2503.14338.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste. Sparsity in deep learning: pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(1), Jan. 2021. ISSN 1532-4435.

K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. ISSN 0893-6080.

B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713, 2018.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 31. Curran Associates, Inc., 2018.

T. Jin, M. Carbin, D. M. Roy, J. Frankle, and G. K. Dziugaite. Pruning's effect on generalization through the lens of training and regularization. In *Advances in Neural Information Processing Systems (NIPS)*, Advances in Neural Information Processing Systems (NIPS), Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

J. Kouchly, B. Finkelshtein, M. Bronstein, and R. Levie. Efficient learning on large graphs using a densifying regularity lemma. *Int. Conf. on Learning Representations (ICLR)*, 2025.

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

Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539.

N. Lee, T. Ajanthan, and P. H. S. Torr. Snip: Single-shot Network Pruning based on Connection Sensitivity. In *Int. Conf. on Learning Representations (ICLR)*. OpenReview.net, 2019.

M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80131-5.

R. Levie. A Graphon-Signal Analysis of Graph Neural Networks. *Advances in Neural Information Processing Systems (NIPS)*, 2023.

L. Lovász. *Large Networks and Graph Limits*, volume 60 of *Colloquium Publications*. American Mathematical Society, 2012.

L. László and B. Szegedy. Szemerédi's Lemma for the Analyst. *Geometric and Functional Analysis*, 17:252–270, 04 2007. doi: 10.1007/s00039-007-0599-6.

C. M. S. Medeiros and G. A. Barreto. An efficient method for pruning the multilayer perceptron based on the correlation of errors. In J. M. de Sá, L. A. Alexandre, W. Duch, and D. Mandic, editors, *Artificial Neural Networks – ICANN 2007*, pages 219–228, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-74690-4.

G. Menghani. Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better. *ACM Computing Surveys*, 55(12), Mar. 2023. ISSN 0360-0300. doi: 10.1145/3578938.

C. Merkwirth and T. Lengauer. Automatic generation of complementary descriptors with molecular graph networks. *Journal of Chemical Information and Modeling*, 45(5):1159–1168, 2005.

H. N. Mhaskar. Neural Networks for Optimal Approximation of Smooth and Analytic Functions. *Neural Computation*, 8(1):164–177, 1996.

D. C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta. Scalable Training of Artificial Neural Networks with Adaptive Sparse Connectivity Inspired by Network Science. *Nature Communications*, 9(1):2383, 2018. doi: 10.1038/s41467-018-04316-3.

E.-A. Peste. *Efficiency and Generalization of Sparse Neural Networks*. Phd thesis, Institute of Science and Technology Austria, 2023.

A. Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8:143–195, 1999. doi: 10.1017/S0962492900002919.

L. Rauchwerger and R. Levie. A Note on Graphon-Signal Analysis of Graph Neural Networks. *arXiv preprint*, 2025.

L. Rauchwerger, S. Jegelka, and R. Levie. Generalization, Expressivity, and Universality of Graph Neural Networks on Attributed Graphs. *Int. Conf. on Learning Representations (ICLR)*, 2025.

E. Rossi, B. Charpentier, F. Di Giovanni, F. Frasca, S. Günnemann, and M. Bronstein. Edge directionality improves learning on heterophilic graphs. In *Proceedings of Machine Learning Research*, 05 2023. doi: 10.48550/arXiv.2305.10498.

U. Tanielian, M. Sangnier, and G. Biau. Approximating lipschitz continuous functions with groupsort neural networks. *International Conference on Artificial Intelligence and Statistics*, abs/2006.05254, 2020.

N. C. Thompson, K. H. Greenewald, K. Lee, and G. F. Manso. The Computational Limits of Deep Learning. *Ninth Workshop on Computing within Limits*, abs/2007.05558, 2020.

K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How Powerful are Graph Neural Networks? *Int. Conf. on Learning Representations (ICLR)*, 2019.

D. Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017. ISSN 0893-6080.

A. Proof of Deep Neural Networks as MPNNs

Lemma 3.2 shows that the output of a B-strongly dense neural network can be represented as the output of an L-layer B-IR-MPNN applied to its induced computational kernel. To prove Lemma 3.2, we first show Lemma A.1, which states that the B-IR-MPNN applied to a computational kernel-signal is stationary on $U^{(\text{bias})}$.

Lemma A.1. *Let* $(K_{(\mathbf{W},\mathbf{b})}, f) \in \mathcal{CK}(B, L, d_0, d_L) \times \mathcal{CS}(L, d_0, d_L)$ *be a computational kernel-signal. Then, the hidden representations of the* B*-IR-MPNN satisfy:*

$$f^{(\ell)}|_{U^{(\text{bias})}} \equiv 1, \quad \text{for} \quad \ell = 0, \dots, L.$$

That is, the hidden representations are constant and share the same value on $U^{(\text{bias})}$.

Proof. We prove the claim using induction. Recall that by the definition of computational kernel for any $v \in U^{(\text{bias})}$:

$$K(v, u) = \begin{cases} (L+2)/B & \text{if} \quad u \in U^{(\text{bias})}, \\ 0 & \text{otherwise.} \end{cases}$$

Induction Base. By the definitions of computational signals and of the B-IR-MPNN, we have

$$f^{(0)}|_{U^{(\text{bias})}} \equiv 1.$$

Induction Assumption. We assume that $f^{(\ell-1)}|_{U^{(\text{bias})}} \equiv 1$ for $0 < \ell \leq L$.

Induction Step. By the definition of the B-IR-MPNN and the induction assumption, for any $v \in U^{(\text{bias})}$:

$$\begin{aligned} f^{(\ell)}(v) &:= B \cdot \text{ReLU}\left(\int_{[0,1]} K(v,u) f^{(\ell-1)}(u) du\right) \\ &= B \cdot \text{ReLU}\left(\int_{U^{(\text{bias})}} (L+2)/B f^{(\ell-1)}(u) du\right) \\ &= (L+2) \cdot \text{ReLU}\left(\int_{U^{(\text{bias})}} f^{(\ell-1)}(u) du\right) \\ &= (L+2) \cdot \text{ReLU}\left(\int_{U^{(\text{bias})}} du\right) \\ &= (L+2) \cdot \mu\left(U^{(\text{bias})}\right) = 1. \end{aligned}$$

This concludes the proof. □

Now, we use Lemma A.1 to show Lemma 3.2.

Lemma A.2. *Let* $(\mathbf{W}, \mathbf{b})$ *be the dense parameters of* $\Theta_{(\mathbf{W},\mathbf{b})} \in \mathcal{NN}_{\text{dense}}(B, L, d_0, d_L)$*. Then, for any input* $\mathbf{x} \in \mathbb{R}^{d_0}$ *and output channel* $i \in [d_L]$*, we have*

$$\Theta_{(\mathbf{W},\mathbf{b})}(\mathbf{x})_i = \Phi_{B,L}\big(K_{(\mathbf{W},\mathbf{b})}, f_{\mathbf{x}}\big)(v),$$

whenever v *is in the* i*th interval of* $\mathcal{C}^{\text{out}}_{d_L}$.

Proof. We now show that for $\ell = 0, \dots, L$ the activation of the ℓth hidden layer $\mathbf{h}^{(\ell)}_i$ in channel $i \in [d_\ell]$ of the network $\Theta_{(\mathbf{W},\mathbf{b})}$ is equal to the hidden B-IR-MPNN representation $f^{(\ell)}(v)$ whenever v is in the ith interval of $\mathcal{P}^{(\ell)}$, when

$$\mathcal{P}^{(\ell)} := \begin{cases} \mathcal{C}^{\text{in}}_{d_0}, & \text{if} \quad \ell = 0, \\ \mathcal{R}^{(\ell)}_d & \text{if} \quad 0 < \ell < L, \\ \mathcal{C}^{\text{out}}_{d_L} & \text{if} \quad \ell = L. \end{cases}$$

Recall that by the definition of the computational kernel, for any v in the ith interval of $\mathcal{P}^{(\ell)}$, $\ell \in [L]$:

$$K(v, u) = \begin{cases} b^{(\ell)}_i / B & \text{if } u \in U^{(\text{bias})} \\ & \quad \text{and} \\ & \quad v \text{ is in the } i\text{th interval of } \mathcal{P}^{(\ell)}, \\ W^{(\ell)}_{i,j} / B & \text{if } u \text{ is in the } j\text{th interval of } \mathcal{P}^{(\ell-1)} \\ & \quad \text{and} \\ & \quad v \text{ is in the } i\text{th interval of } \mathcal{P}^{(\ell)}, \\ 0 & \text{otherwise.} \end{cases}$$

We prove the statement inductively.

Induction Base. Let $\ell = 0$ and $i \in [d_0]$, then, for any v in the ith interval of $\mathcal{P}^{(0)}$ and $\mathbf{x} = (x_1, \dots, x_{d_0}) \in \mathbb{R}^{d_0}$

$$f^{(0)}(v) = f_{\mathbf{x}}(v) = x_i = \mathbf{h}^{(0)}_i.$$

Induction Assumption. For layers $\ell = 1, \dots, L$, channels $i \in [d_{\ell-1}]$ and for any v in the ith interval of $\mathcal{P}^{(\ell-1)}$, assume that

$$f^{(\ell-1)}(v) = \mathbf{h}^{(\ell-1)}_i.$$

Induction Step. Denote the jth interval of the partition $\mathcal{P}^{(\ell)}$ by $I^{(\ell)}_j$. Then, for any v in the ith interval of the partition $\mathcal{P}^{(\ell)}$:

$$\begin{aligned} f^{(\ell)}(v) &:= B \cdot \text{ReLU}\left(\int_{[0,1]} K(v,u) f^{(\ell-1)}(u) du\right) \\ &= B \cdot \text{ReLU}\Bigg(\sum_{I^{(\ell-1)}_j \in \mathcal{P}^{(\ell-1)}} \int_{I^{(\ell-1)}_j} (W^{(\ell)}_{i,j}/B) f^{(\ell-1)}(u) du \\ &\qquad + \int_{U^{(\text{bias})}} (b^{(\ell)}_i / B) f^{(\ell-1)}(u) du\Bigg) \\ &= \text{ReLU}\Bigg(\sum_{I^{(\ell-1)}_j \in \mathcal{P}^{(\ell-1)}} \int_{I^{(\ell-1)}_j} W^{(\ell)}_{i,j} f^{(\ell-1)}(u) du \\ &\qquad + \int_{U^{(\text{bias})}} b^{(\ell)}_i f^{(\ell-1)}(u) du\Bigg) =: (*) \end{aligned}$$

By the induction assumption $f^{(\ell-1)}(v) = \mathbf{h}^{(\ell-1)}_i$ on any v in the ith interval of $\mathcal{P}^{(\ell-1)}$ and since by definition

$f_{\mathbf{x}}|_{U^{(\text{bias})}} \equiv 1$, then, $f^{(\ell-1)}|_{U^{(\text{bias})}} \equiv 1$ by Lemma A.1 and. Thus

$$\begin{aligned}(*) &= \text{ReLU}\Bigg(\sum_{j=1}^{d_{\ell-1}} \int_{I_j^{(\ell-1)}} W_{i,j}^{(\ell)} \mathbf{h}_j^{(\ell-1)} du \\ &\qquad + \int_{U^{(\text{bias})}} b_i^{(\ell)} du\Bigg) \\ &= \text{ReLU}\Bigg(\sum_{j=1}^{d_{\ell-1}} W_{i,j}^{(\ell)} \mathbf{h}_j^{(\ell-1)} \cdot \mu\left(I_j^{(\ell-1)}\right) \\ &\qquad + b_i^{(\ell)} \mu\left(U^{(\text{bias})}\right)\Bigg) =: (**)\end{aligned}$$

Recall that $\mu\left(U^{(\text{bias})}\right) = 1/(L+2)$ and $\mu\left(I_j^{(\ell-1)}\right) = 1/(d_{\ell-1}(L+2))$ by the definition of the computational kernel. Therefore

$$\begin{aligned}(**) &= \text{ReLU}\Bigg(\sum_{j=1}^{d_{\ell-1}} W_{i,j}^{(\ell)} \mathbf{h}_j^{(\ell-1)} \int_{I_j^{(\ell-1)}} du \\ &\qquad + b_i^{(\ell)} \int_{U^{(\text{bias})}} du\Bigg) \\ &= \text{ReLU}\Bigg(\sum_{j=1}^{d_{\ell-1}} W_{i,j}^{(\ell)} \mathbf{h}_j^{(\ell-1)} \int_{I_j^{(\ell-1)}} du + \frac{1}{L+2} b_i^{(\ell)}\Bigg) \\ &= \text{ReLU}\Bigg(\frac{1}{d_{\ell-1}(L+2)} \sum_{j=1}^{d_{\ell-1}} \left(W_{ij}^{(\ell)} \mathbf{h}_j^{(\ell-1)} + b_i^{(\ell)}\right)\Bigg) \\ &= \mathbf{h}_i^{(\ell)}.\end{aligned}$$

This concludes the proof since

$$\Theta_{(\mathbf{W},\mathbf{b})}(\mathbf{x})_i = \mathbf{h}_i^{(L)} = f^{(L)}(v) = \Phi_{B,L}\big(K_{(\mathbf{W},\mathbf{b})}, f_{\mathbf{x}}\big)(v),$$

whenever v is in the ith interval of $\mathcal{C}_{d_L}^{\text{out}}$

□

B. Proofs of Model Compression

Here we prove our Lipschitz continuity of B-IR-MPNN (Theorem 4.1), our weak regularity lemma (Theorem 4.2), and our compression result (Corollary 4.3).

Lipschitz Continuity of MPNNs. MPNNs were shown to be Lipschitz continuous with respect to the cut norm in (Levie, 2023), with a further extension in (Rauchwerger and Levie, 2025). In this work, we adapt the techniques developed in (Levie, 2023) to our setting in order to obtain an explicit bound on the Lipschitz constant.

We first recall some useful lemmas.

Lemma B.1 ((Levie, 2023), Equation (9)). *Let $f : [0,1] \to \mathbb{R}$ be measurable. Then*

$$\frac{1}{2}\|f\|_1 \le \|f\|_\square \le \|f\|_1.$$

Denote by $L^+[0,1]$ the space of measurable functions $f : [0,1] \to [0,1]$.

Lemma B.2 ((Levie, 2023), Lemma F.5/(Lovász, 2012), Lemma 8.10). *Let $Q : [0,1]^2 \to \mathbb{R}$ be measurable. Then*

$$\|Q\|_\square = \sup_{f,g \in L^+[0,1]} \left| \int_{[0,1]^2} f(x)Q(x,y)g(y)\mathrm{d}x\mathrm{d}y \right|,$$

where the supremum is attained for some $f, g \in L^+[0,1]$ with values in $\{0,1\}$.

Lemma B.3 (Cut Norm Lipschitz Continuity). *Let $B \in \mathbb{R}$ and $\Phi_{B,L}$ be the L-layer B-IR-MPNN and $(K,f),(J,g)$ be two attributed Kernels. Then,*

$$\begin{aligned}&\|\Phi_{B,L}(K,f) - \Phi_{B,L}(J,g)\|_\square \\ &\qquad \le 2^L B^L (\|f-g\|_\square \|K-J\|_\square).\end{aligned}$$

Proof. Recall that for a kernel-signal (T,q)

$$\Phi_{B,L}(T,q)(x) := q^{(L)}(x),$$

and for $\ell \in [L]$,

$$q^{(\ell)}(x) := B \cdot \text{ReLU}\left(\int_{[0,1]} T(x,y) q^{(\ell-1)}(y)\,\mathrm{d}y\right).$$

We first show that for every

$$\ell, \|f^{(\ell)}\|_\infty, \|g^{(\ell)}\|_\infty \le B^\ell.$$

Indeed, for any $x \in [0,1]$

$$|f^{(\ell)}(x)| \le B \left| \int_{[0,1]} K(x,y) f^{(\ell-1)}(y)\mathrm{d}y \right|.$$

Hence, by Hölder's inequality

$$|f^{(\ell)}(x)| \le B\|f^{(\ell-1)}\|_1 \le B\|f^{(\ell-1)}\|_\infty,$$

so we also have

$$\|f^{(\ell)}\|_\infty \le B\|f^{(\ell-1)}\|_\infty.$$

By solving a recurrence sequence, and using $\|f^{(0)}\|_\infty \le 1$, we get $\|f^{(\ell)}\|_\infty \le B^\ell$.

Next, we consider

$$f^{(\ell)}(x) - g^{(\ell)}(x) = B\left(\int K(x,y) f^{(\ell-1)}(y)\,\mathrm{d}y - \int J(x,y) g^{(\ell-1)}(y)\,\mathrm{d}y\right),$$

and decompose this difference as

$$\int K(x,y) f^{(\ell-1)}(y)\,\mathrm{d}y - \int J(x,y) g^{(\ell-1)}(y)\,\mathrm{d}y = \int J(x,y)\big(f^{(\ell-1)}(y) - g^{(\ell-1)}(y)\big)\,\mathrm{d}y + \int \big(K(x,y) - J(x,y)\big) f^{(\ell-1)}(y)\,\mathrm{d}y.$$

By the triangle inequality,

$$\|\Phi_{B,\ell}(K,f) - \Phi_{B,\ell}(J,g)\|_\square \le B\|\int J(x,y)(f^{(\ell-1)}(y) - g^{(\ell-1)}(y))\,\mathrm{d}y\|_\square \quad (4)$$
$$+ B\|\int (K-J)(x,y) f^{(\ell-1)}(y)\,\mathrm{d}y\|_\square. \quad (5)$$

We bound Equation (4), using Lemma B.1, by

$$\begin{aligned}\|\int J(x,y)(f^{(\ell-1)}(y) - g^{(\ell-1)}(y))\,\mathrm{d}y\|_\square &\le \|\int J(x,y)(f^{(\ell-1)}(y) - g^{(\ell-1)}(y))\,\mathrm{d}y\|_1 \\ &\le \int \Big|\int J(x,y)(f^{(\ell-1)}(y) - g^{(\ell-1)}(y))\,\mathrm{d}y\Big| dx \\ &\le \int\int \Big|J(x,y)(f^{(\ell-1)}(y) - g^{(\ell-1)}(y))\Big| \mathrm{d}y\mathrm{d}x.\end{aligned}$$

Hence, by Hölder's inequality,

$$\begin{aligned}\|\int J(x,y)(f^{(\ell-1)}(y) - g^{(\ell-1)}(y))\,\mathrm{d}y\|_\square &\le \int 1 \int \Big|(f^{(\ell-1)}(y) - g^{(\ell-1)}(y))\Big| \mathrm{d}y dx \\ &\le \|f^{(\ell-1)} - g^{(\ell-1)}\|_1 \\ &\le 2\|f^{(\ell-1)} - g^{(\ell-1)}\|_\square\end{aligned}$$

where the last inequality is due to Lemma B.1.

We bound Equation (5) as follows. Let $q = f^{(\ell-1)}/B^{\ell-1}$. By the application of ReLU, and by the bound $\|f^{(\ell-1)}\|_\infty \le B^{\ell-1}$, the function $f^{\ell-1}$ maps $[0,1]$ to $[0, B^{\ell-1}]$, so $q : [0,1] \to [0,1]$. Let $Q := K - J : [0,1]^2 \to [-2,2]$. Our goal is to bound $\|\int Q(x,y)q(y)dy\|_\square$. Let $S \subset [0,1]$ be the set for which the cut norm is realized. The existence of such an S is guaranteed by Lemma B.2. Hence,

$$\|\int Q(x,y)q(y)dy\|_\square = \int\int \mathbb{1}_S(x) Q(x,y) q(y) dy dx.$$

Using Lemma B.2, we get

$$|\int\int \mathbb{1}_S(x) Q(x,y) q(y) dy dx| \le \|Q\|_\square.$$

Combining the bounds, we obtain

$$\|f^{(\ell)} - g^{(\ell)}\|_\square \le 2B\|f^{(\ell-1)} - g^{(\ell-1)}\|_\square + B^\ell \|K - J\|_\square.$$

Hence,

$$\|f^{(\ell)} - g^{(\ell)}\|_\square \le 2B\|f^{(\ell-1)} - g^{(\ell-1)}\|_\square + B^\ell \|K - J\|_\square.$$

Now, denote by $c^{(\ell)}$ bounds on $\|f^{(\ell)} - g^{(\ell)}\|_\square$, for $\ell = 1, \ldots, L$, which satisfy

$$c^{(\ell)} = 2Bc^{(\ell-1)} + B^\ell r,$$

where $r = \|K - J\|_\square$. Solving this recurrence sequence, we get

$$c^{(\ell)} = 2^\ell B^\ell c^{(0)} + \sum_{k=0}^{\ell-1} 2^k B^k r \le 2^\ell B^\ell (c^{(0)} + r).$$

Hence,

$$\|f^{(\ell)} - g^{(\ell)}\|_\square \le 2^\ell B^\ell (\|f^{(0)} - g^{(0)}\|_\square + \|K - J\|_\square).$$

□

The following result is a consequence of Lemma B.3.

Theorem 4.1 (Computational Cut Distance Lipschitz Continuity)**.** *Let $B > 0$ and let $\Phi_{B,L}$ be the L-layer B-IR-MPNN. Let $(K,f),(J,g) \in \mathcal{CK}(B,L,d_0,d_L) \times \mathcal{CS}(L,d_0,d_L)$ be two computational kernel-signals. Let*

$$\mathrm{Out}_{K,f} := \mathbb{1}_{U^{(L)}} \Phi_{B,L}(K,f),$$
$$\mathrm{Out}_{J,g} := \mathbb{1}_{U^{(L)}} \Phi_{B,L}(J,g).$$

Then

$$\|\mathrm{Out}_{K,f} - \mathrm{Out}_{J,g}\|_\square \;\le\; 2^L B^L\, \delta_\square^{\mathrm{comp}}\big((K,f),(J,g)\big).$$

Proof. Consider an arbitrary measure preserving bijection $\phi \in \mathcal{S}_L$ that preserves the layer structure. Applying Lemma B.3 to the pair (K,f) and $(J^\phi, g\circ\phi)$ yields

$$\|\Phi_{B,L}(K,f) - \Phi_{B,L}(J^\phi, g\circ\phi)\|_\square \le 2^L B^L \big(\|K - J^\phi\|_\square + \|f - g\circ\phi\|_\square\big).$$

Since the above inequality holds for every $\phi \in \mathcal{S}_L$, taking the ϕ that minimizes the right-hand-side gives

$$\|\Phi_{B,L}(K,f) - \Phi_{B,L}(J^\phi, g\circ\phi)\|_\square \quad (6)$$
$$\le 2^L B^L\, \delta_\square^{\mathrm{comp}}\big((K,f),(J,g)\big). \quad (7)$$

Denote $h = \Phi_{B,L}(K,f) - \Phi_{B,L}(J^\phi, g\circ\phi)$. Note that the signal cut norm in (6) is either realized on the set S of point where h is positive, or the set of points where it is negative. Hence, multiplying h by $\mathbb{1}_{U^{(L)}}$ can only decrease the cut norm of h, as the positive and negative supports of $\mathbb{1}_{U^{(L)}}h$ are subsets of the positive and negative supports of h. Therefore,

$$\|\mathbb{1}_{U^{(L)}}\Phi_{B,L}(K,f) - \mathbb{1}_{U^{(L)}}\Phi_{B,L}(J^\phi, g\circ\phi)\|_\square \leq 2^L B^L \delta_\square^{\text{comp}}\big((K,f),(J,g)\big).$$

Lastly, note that by definition, any measure preserving bijection $\phi \in \mathcal{S}_L$ keeps $U^{(L)}$ unchanged, so

$$\|\mathbb{1}_{U^{(L)}}\Phi_{B,L}(K,f) - \mathbb{1}_{U^{(L)}}\Phi_{B,L}(J^\phi, g\circ\phi)\|_\square = \|\text{Out}_{K,f} - \text{Out}_{J,g}\|_\square,$$

which completes the proof. □

The Regularity Lemma for computational Kernels. Let us recall some definitions. Let μ denote the Lebesgue measure on $[0,1]$. A *partition* of $[0,1]$ is a sequence $\mathcal{P}_k = \{P_1,\dots,P_k\}$ of disjoint measurable sets such that $\bigcup_{j=1}^k P_j = [0,1]$. The partition is called an *equipartition* if $\mu(P_i) = \mu(P_j)$ for all $i,j \in [k]$. We write $\mathbb{1}_S$ for the indicator function of a set S. We define step functions as follows.

Definition B.4. Given a partition $\mathcal{P}_n$, define the space $\mathcal{S}^2_{\mathcal{P}_n}$ of *step functions* $[0,1]^2 \mapsto \mathbb{R}$ over the partition $\mathcal{P}_n$ to be the space of functions of the form

$$F(x,y) = \sum_{i,j\in[n]\times[n]} c_{i,j}\mathbb{1}_{P_i\times P_j}(x,y),$$

for any choice of $\{c_{i,j}\in\mathbb{R}\}_{i,j\in[n]\times[n]}$.

Denote the space of kernels $K:[0,1]^2\to[-1,1]$ by $\mathcal{W}_1$. Notice that any step kernel with respect to $\mathcal{P}_n$ is in the intersection $\mathcal{S}^2_{\mathcal{P}_n}\cap\mathcal{W}_1$. We define the projection of a kernel onto a partition as its blockwise average over the parts of the partition.

Definition B.5. Let $\mathcal{P}_n = \{P_1,\dots,P_n\}$ be a partition of $[0,1]$, and $K\in\mathcal{W}_1$ be a kernel. The *projection* of K upon $\mathcal{S}^2_{\mathcal{P}_n}$ is the step kernel $K_{\mathcal{P}_n}$ that attains the value

$$K_{\mathcal{P}_n}(x,y) = \frac{1}{\mu(P_i)\mu(P_j)}\int_{[0,1]^2} K(x,y)\mathbb{1}_{P_i\times P_j}(x,y)\mathrm{d}x\mathrm{d}y,$$

for every $(x,y)\in P_i\times P_j$ and $1\leq i,j\leq n$.

The projection is also called the *stepping operator*. The following theorem, provided in (Rauchwerger and Levie, 2025), Appendix B, is an adaptation of the "analyst's version" of the *weak regularity lemma* for kernels, originally introduced in (László and Szegedy, 2007).

Theorem B.6 ((Rauchwerger and Levie, 2025), Theorem B.5)**.** *Let $\epsilon>0$. For every kernel $K\in\mathcal{W}_1$ there exists a partition $\mathcal{P}_k$ of $[0,1]$ into $k = 2^{2^{\lceil 1/\epsilon^2\rceil}}$ sets, and a step kernel $K_k\in\mathcal{S}^2_{\mathcal{P}_k}\cap\mathcal{W}_1$, such that*

$$\|K - K_k\|_\square \leq \epsilon.$$

The following lemma is given in (Rauchwerger and Levie, 2025) as Lemma B.10.

Lemma B.7. *Let $\mathcal{P}_n = \{P_1,\dots,P_n\}$ be a partition of $[0,1]$, and Let $V,R\in\mathcal{S}^2_{\mathcal{P}_n}\cap\mathcal{W}_1$. Then, the supremum of*

$$\sup_{S,T\subset[0,1]}\left|\int_S\int_T \big(V(x,y)-R(x,y)\big)dxdy\right|$$

is attained for S,T of the form

$$S = \bigcup_{i\in s} P_i\,,\quad T = \bigcup_{j\in t} P_j,$$

where $t,s\subset[n]$.

As part of the proof of Theorem 4.2, we first show that any computational kernel can be approximated by a coarse step kernel, where the partition underlying the coarse kernel is not an equipartition in general. Hence, the last step of the proof involves approximating the coarse kernel by one based on an equipartition. For this, we recall an equatizing lemma from (Levie, 2023). Inspecting the proof of Lemma B.1 from (Levie, 2023), we can reformulate the lemma as follows.

Lemma B.8 ((Levie, 2023), Equitizing partition)**.** *Let $\mathcal{P}_k$ be a partition of $[a,b]$ into k sets (generally not of the same measure). Then, for any $n>k$ there exists an equipartition $\mathcal{E}_n$ of $[0,1]$ into n sets which satisfies the following. For each $j\in[k]$, let $\mathcal{E}_n^j$ be the set of all parts of $\mathcal{E}_n$ that are subsets of $P_j\in\mathcal{P}_k$. Denote*

$$\mathcal{T}_n := \bigcup_{j=1}^k \mathcal{E}_n^j,$$

and $\mathcal{R}_n = \mathcal{E}_n\setminus\mathcal{T}_n$. Then, there exists an integer $h\leq k$ such that $\mu(\cup\mathcal{R}_n) = h/n$ and $\mu(\cup\mathcal{T}_n) = 1-h/n$. We call $\mathcal{T}_n$ the refinement parts *of $\mathcal{E}_n$ and $\mathcal{R}_n$ the* remainder parts.

For completeness, we repeat the proof.

Proof. Let $\mathcal{P}_k = \{P_1,\dots,P_k\}$ be a measurable partition of $[0,1]$ and let $n>k$. For each $i\in[k]$, subdivide P_i into measurable sets $P_{i,1},\dots,P_{i,m_i}$ as follows. If $\mu(P_i)<1/n$, set $m_i=1$ and $P_{i,1}=P_i$. Otherwise, choose $m_i\geq 2$ such that $\mu(P_{i,j}) = \frac{1}{n}$ for $j\in[m_i-1]$ and $\mu(P_{i,m_i})\leq\frac{1}{n}$. We refer to P_{i,m_i} as the remainder of P_i. Define the sequence of parts of measure $1/n$ to be

$$\mathcal{Q} := \{P_{1,1},\dots,P_{1,m_1-1},P_{2,1},\dots,P_{2,m_2-1},\dots,P_{k,1},\dots,P_{k,m_k-1}\},$$

where for indices i with $m_i = 1$ there is no contribution in $\mathcal{Q}$. Let $l := |\mathcal{Q}|$. By construction, $\mu\Big(\bigcup \mathcal{Q}\Big) = \frac{l}{n}$. Let $\Pi := \bigcup_{i=1}^{k} P_{i,m_i}$. Then $\mu(\Pi) = 1 - \frac{l}{n} = \frac{h}{n}$, for $h := n - l$. Since each P_{i,m_i} has measure at most $1/n$ and there are at most k such sets, we have $h \leq k$.

Next, partition Π into h measurable sets $\Pi_1, \dots, \Pi_h$ each of measure exactly $1/n$. Define

$$\mathcal{E}_n := \mathcal{Q} \cup \{\Pi_1, \dots, \Pi_h\}.$$

Then $\mathcal{E}_n$ is a partition of $[0,1]$ into n sets, each of measure $1/n$. Let

$$\mathcal{T}_n := \mathcal{Q}, \qquad \mathcal{R}_n := \{\Pi_1, \dots, \Pi_h\}.$$

By construction,

$$\mu\Big(\bigcup \mathcal{T}_n\Big) = 1 - \frac{h}{n}, \qquad \mu\Big(\bigcup \mathcal{R}_n\Big) = \frac{h}{n}, \qquad h \leq k,$$

and each element of $\mathcal{T}_n$ is contained in some $P_i \in \mathcal{P}_k$. □

The following lemma is a basic result from measure theory.

Lemma B.9. *For any partition $\mathcal{P}$ of $[a,b]$, there is a measure preserving bijection ϕ (between co-null sets of $[0,1]$) that maps $\mathcal{P}$ into an interval partition $\mathcal{J}$ of $[a,b]$. Namely, every $J \in \mathcal{J}$ is of the form $J = \phi^{-1}(P)$, up to a null-set, for some $P \in \mathcal{P}$.*

Proof. First, let $[a,b] = [0,1]$. Let $\mathcal{P} = \{P_1, P_2, \dots\}$ be the partition of $[0,1]$ into measurable, disjoint sets, with $\sum_i \mu(P_i) = 1$. For each i, let $q_i = \mu(P_i)$. Define intervals

$$\begin{aligned} J_1 &= [0, q_1], \\ J_2 &= [q_1, q_1 + q_2], \\ &\dots, \\ J_n &= \Big[\sum_{k=1}^{n-1} q_k, \sum_{k=1}^{n} q_k\Big]. \end{aligned}$$

This gives an interval partition $\mathcal{J} = \{J_1, J_2, \dots\}$ of $[0,1]$. For any two measurable sets $A, B \subset [0,1]$ of equal measure, there exists a measure-preserving bijection $f : A \to B$, up to null sets. Applying this to each P_i and J_i, we obtain bijections $\phi_i : J_i \to P_i$ which are measure-preserving (up to null sets). Define $\phi(x) = \phi_i(x)$ for $x \in J_i$. Then $\phi : [0,1] \to [0,1]$ is a measure-preserving bijection up to null sets. Moreover, by construction, each $J_i = \phi^{-1}(P_i)$ up to a null set. If the original interval is $[a,b] \neq [0,1]$, let $T : [0,1] \to [a,b]$ be the affine map $T(x) = a + (b-a)x$. Then the map $\tilde{\phi} := T \circ \phi \circ T^{-1} : [a,b] \to [a,b]$ is a measure-preserving bijection (up to null sets) that maps $\mathcal{P}$ to an interval partition of $[a,b]$, completing the proof. For any measurable set $B \subset [a,b]$, let $A := T^{-1}(B) \subset [0,1]$. Then $\tilde{\phi}^{-1}(B) = T(\phi^{-1}(A))$, and the measure on $[a,b]$ satisfies $\mu_{[a,b]}(\tilde{\phi}^{-1}(B)) = (b-a)\,\mu(\phi^{-1}(A)) = (b-a)\,\mu(A) = \mu_{[a,b]}(B)$, so $\tilde{\phi}$ is measure-preserving. Furthermore, by construction, $\tilde{\phi}$ maps $\mathcal{P}$ to an interval partition of $[a,b]$, completing the proof. □

The proof strategy of Theorem 4.2 uses techniques from (Levie, 2023), Corollary B.11 and (Rauchwerger and Levie, 2025), Theorem 4.9, both extending techniques from (László and Szegedy, 2007). We begin by applying the weak regularity lemma for kernels (Theorem B.6) to obtain an initial approximation of the graph-induced kernel. We then project this kernel onto the partition provided by Lemma B.8, yielding a structured kernel approximation that corresponds to a computational graph.

Theorem 4.2. *Let $K \in \mathcal{CK}(B, L, d_0, d_L)$ be an L-layer computational kernel. For any $\epsilon > 0$, there exists an L-layer computational kernel $K' \in \mathcal{CK}(B, L, d_0, d_L)$ with hidden dimension*

$$d = 8ML \left\lceil \frac{2^{2\lceil 16/\epsilon^2 \rceil}}{\epsilon} \right\rceil,$$

where M is the least common multiple of d_0 and d_L, such that $\delta_{\square}^{\mathrm{comp}}(K, K') < \epsilon$.

Throughout the proof, we ensure that the structure of a partition underlying a computational kernel is preserved under the application of the weak regularity lemma.

Proof. Let $\epsilon > 0$, and let K be a computational kernel based on the partition $\mathcal{Q}$. First, apply Theorem B.6 to the computational kernel K with accuracy $\epsilon/4$. This yields a step kernel K_k, and a measurable partition $\mathcal{P}_k$ of $[0,1]$, such that

$$k = 2^{2\lceil 16/\epsilon^2 \rceil}, \qquad \|K - K_k\|_{\square} \leq \epsilon/4.$$

The partition $\mathcal{P}_k$ is a general measurable partition (not necessarily an interval partition, and not respecting the structure of a computational kernel).

Let $\mathcal{L}_r$ be the partition of $[0,1]$, with $r = L + d_0 + d_L$ which is the union of:

- all intermediate layers $\{U^{(1)}, \dots, U^{(L-1)}\}$,
- the bias layer $\{U^{(\mathrm{bias})}\}$,
- the coarse input partition $\mathcal{C}_{d_0}^{\mathrm{in}}$,
- the coarse output partition $\mathcal{C}_{d_L}^{\mathrm{out}}$.

We call $\mathcal{L}_r$ the *layer partition*. Let $\mathcal{P}$ be the coarsest common refinement of $\mathcal{P}_k$ and $\mathcal{L}_r$. We have

$$|\mathcal{P}| \leq |\mathcal{P}_k|\,|\mathcal{L}_r| = 2^{2\lceil 16/\epsilon^2 \rceil}(L + d_0 + d_L),$$

and $\mathcal{P}$ is a measurable (not interval) partition.

By Lemma B.7, the cut norm $\|K_k - K_{\mathcal{P}}\|_\square$ is attained on sets of the form

$$S = \bigcup_{P_i \in s} P_i, \qquad T = \bigcup_{P_j \in t} P_j,$$

for some $s, t \subset \mathcal{P}$. Using the definition of the projected kernel,

$$\begin{aligned}\|K_k - K_{\mathcal{P}}\|_\square &= \left| \int_S \int_T (K_k - K_{\mathcal{P}})(x,y)\, dx\, dy \right| \\ &= \left| \int_S \int_T (K_k - K)(x,y)\, dx\, dy \right| \\ &= \|K_k - K\|_\square.\end{aligned}$$

Therefore,

$$\|K - K_{\mathcal{P}}\|_\square \leq \|K - K_k\|_\square + \|K_k - K_{\mathcal{P}}\|_\square \leq \epsilon/2.$$

Note that the parts of $\mathcal{P}_k$ that intersect $U^{(0)}$ constitute a refinement of $\mathcal{C}^{\text{in}}_{d_0}$. Similarly, the parts of $\mathcal{P}$ that intersect $U^{(L)}$ constitute a refinement of $\mathcal{C}^{\text{out}}_{d_L}$. Note as well that K is constant on every set of the form $C \times D$ or $D \times C$ where C is a part in $\mathcal{C}^{\text{in}}_{d_0}$ or $\mathcal{C}^{\text{out}}_{d_L}$ and D is any part of $\mathcal{Q}$ (the partition underlying K) that intersect (or equivalently, is a subset of) $U^{(1)} \cup \ldots \cup U^{(L-1)} \cup U^{(\text{bias})}$.

Hence, by the definition of projection and since $\mathcal{P}$ refines $\mathcal{L}_r$, the kernel $K_{\mathcal{P}}$ has the same property: it is constant on every set of the form $C \times D$ or $D \times C$ where C is a part in $\mathcal{C}^{\text{in}}_{d_0}$ or $\mathcal{C}^{\text{out}}_{d_L}$ and D is any part of $\mathcal{P}$ that intersect $U^{(1)} \cup \ldots \cup U^{(L-1)}$. We define the partition $\mathcal{P}'$ as the partition containing as parts the intervals of $\mathcal{C}^{\text{in}}_{d_0}$ and $\mathcal{C}^{\text{out}}_{d_L}$ and all parts of $\mathcal{P}$ outside $\mathcal{C}^{\text{in}}_{d_0}$ and $\mathcal{C}^{\text{out}}_{d_L}$. By the above analysis, $K_{\mathcal{P}}$ is a step graphon with respect to $\mathcal{P}'$.

For each $\ell = 1, \ldots, L-1$, let $\mathcal{P}^{(\ell)}$ be the set of parts of $\mathcal{P}'$ that intersect (or equivalently are subsets of) $U^{(\ell)}$. Similarly define $\mathcal{P}^{(\text{bias})}$. Note that $\mathcal{P}^{(\ell)}$ is a partition of $U^{(\ell)}$ and $\mathcal{P}^{(\text{bias})}$ a partition of $U^{(\text{bias})}$. Note as well that

$$|\mathcal{P}^{(\text{bias})}|, |\mathcal{P}^{(\ell)}| \leq 2^{2^{\lceil 16/\epsilon^2 \rceil}}(L + d_0 + d_L).$$

Let $M \in \mathbb{N}$ be the lowest common multiple of d_0 and d_L. Let $n = C(L+2)M$ for some $C \in \mathbb{N}$ to be specified later. Now, apply the equatizing partition lemma (Lemma B.8), to equatize $\mathcal{P}^{(\ell)}$ into CM parts $\mathcal{E}^{(\ell)}_{CM}$, each of measure $1/n$. Similarly, define $\mathcal{E}^{(\text{bias})}_{CM}$. Here, we restrict the choice of C to satisfy

$$CM > \max_{\ell = 1, \ldots, L-1, (\text{bias})} |\mathcal{P}^{(\ell)}|.$$

By the equatizing lemma, there exist numbers $h^{(\ell)} \leq |\mathcal{P}^{(\ell)}|$ such that the first $CM - h^{(\ell)}$ parts of $\mathcal{E}^{(\ell)}_{CM}$ are the refinement parts $\mathcal{T}^{(\ell)}_{CM}$, and the last $h^{(\ell)}$ parts of $\mathcal{E}^{(\ell)}_{CM}$ are the remainder parts $\mathcal{R}^{(\ell)}_{CM}$. Similarly, we consider $h^{(\text{bias})} < |\mathcal{P}^{(\text{bias})}|$, $\mathcal{T}^{(\text{bias})}_{CM}$ and $\mathcal{R}^{(\text{bias})}_{CM}$.

Denote $\mathcal{E}_j = \cup_\ell \mathcal{E}^{(\ell)}_{CM} \cup \mathcal{E}^{(\text{bias})}_{CM} \cup \mathcal{C}^{\text{in}}_{d_0} \cup \mathcal{C}^{\text{out}}_{d_L}$, where $j = |\mathcal{E}_j| = CML + d_0 + d_L$. Denote $\mathcal{R} = \cup_\ell \mathcal{R}^{(\ell)}_{CM} \cup \mathcal{R}^{(\text{bias})}_{CM}$ and $R = \cup \mathcal{R}$. By construction, the projection $K_{\mathcal{E}_j}$ of K upon $\mathcal{E}_j$ is equal to $K_{\mathcal{P}}$ for every (x,y) outside the set $([0,1] \times R) \cup (R \times [0,1])$. Note that

$$\begin{aligned}&\mu(([0,1] \times R) \cup (R \times [0,1])) \\ &\qquad \leq 2\big(h^{(\text{bias})} + \sum_\ell h^{(\ell)}\big)/n \leq 2L|\mathcal{P}'|/n.\end{aligned}$$

Moreover, the pointwise distance between $K_{\mathcal{P}}$ and $K_{\mathcal{E}_j}$ for any $(x,y) \in ([0,1] \times R) \cup (R \times [0,1])$ is bounded by 2. Hence,

$$\|K_{\mathcal{P}} - K_{\mathcal{E}_j}\|_\square \leq \|K_{\mathcal{P}} - K_{\mathcal{E}_j}\|_1 \leq 4L|\mathcal{P}'|/n.$$

Hence, by the triangle inequality

$$\|K - K_{\mathcal{E}_j}\|_\square \leq \epsilon/2 + 4L|\mathcal{P}'|/n.$$

We retroactively choose n such that $4L|\mathcal{P}'|/n \leq \epsilon/2$. For this, we choose

$$n = 8ML(L+2)\left\lceil \frac{2^{2^{\lceil 16/\epsilon^2 \rceil}}}{\epsilon} \right\rceil,$$

which leads to

$$\begin{aligned}4L|\mathcal{P}'|/n &\leq 4L2^{2^{\lceil 16/\epsilon^2 \rceil}}(L + d_0 + d_L)/n \\ &\leq 4L2^{2^{\lceil 16/\epsilon^2 \rceil}}(L + 2M)/n \\ &\leq (L+2M)\epsilon/2(ML + 2M) \leq \epsilon/2,\end{aligned}$$

while still n being of the form $C(L+2)M$ for some $C \in \mathbb{N}$

Next, we map the equipartition $\mathcal{E}_j$ into an interval equipartition which underlies a computational kernel, i.e., which refines $\mathcal{L}_r$. For this, using Lemma B.9, for each $\ell \in \{1, \ldots, L-1, (\text{bias})\}$ apply a measure preserving bijection $\phi^{(\ell)}$ mapping $\mathcal{E}^{(\ell)}_{CM}$ into the interval equipartition $\mathcal{I}^{(\ell)}_{CM}$ of $U^{(\ell)}$. Denote

$$\mathcal{J} = \mathcal{C}^{\text{in}}_{d_0} \cup \mathcal{I}^{(1)}_{CM} \cup \ldots \cup \mathcal{I}^{(L-1)}_{CM} \cup \mathcal{I}^{(\text{bias})}_{CM} \cup \mathcal{C}^{\text{out}}_{d_L},$$

and note that $\mathcal{J}$ is an interval equipartition of $[0,1]$ into n parts underlying a computational kernel, i.e., refines $\mathcal{L}_r$. Consider the corresponding measure preserving bijection ϕ over $[0,1]$, defined by

$$\phi(x) = \begin{cases} \phi^{(\ell)}(x) & x \in U^{(\ell)}, \text{ for } \ell \in [L-1] \cup \{(\text{bias})\}, \\ x & \text{otherwise.} \end{cases}$$

Consider the kernel $K^\phi(x,y) = K(\phi(x),\phi(y))$, and note that $(K^\phi)_\mathcal{J}$ is a computational kernel in $\mathcal{CK}(B,L,d_0,d_L)$ with hidden dimension

$$d = 8ML\left\lceil \frac{2^{2^{\lceil 16/\epsilon^2\rceil}}}{\epsilon}\right\rceil.$$

By the above construction, this kernel satisfies the commutation property between the measure-preserving bijection and the projection

$$(K^\phi)_\mathcal{J} = (K_{\mathcal{E}_j})^\phi.$$

This directly leads to

$$\delta_\square^{\text{comp}}(K, K_\mathcal{J}) \le \epsilon.$$

□

Network Architecture Approximation. We now use Theorem 4.1 and Theorem 4.2 to prove Corollary 4.3. The proof proceeds by first enlarging the computational graph output dimension by adding unconnected computational units to the output layer and inducing it into a kernel. We then establish a computational kernel approximation via Theorem 4.2. Lastly, we use the Lipschitz continuity of graph network of Theorem 4.1, to bound the output.

Corollary B.10 (Network Architecture Approximation)**.** *Let $L, d_0, d_L \in \mathbb{N}$ with $L \ge 2$, let $B \ge L+2$, and $\epsilon > 0$. Then, for any network $\Theta \in \mathcal{NN}_{\text{dense}}(B,L,d_0,d_L)$ there is a network $\Theta' \in \mathcal{NN}_{\text{dense}}(B,L,d_0,d_L)$ with hidden dimension*

$$d = 8ML\left\lceil \frac{1}{\epsilon}\cdot 2^{2^{\left\lceil \frac{16((L+2)d_L)^2(2B)^{2L}}{\epsilon^2}\right\rceil}}(L+2)d_L(2B)^L\right\rceil,$$

such that $\|\Theta - \Theta'\|_\infty < \epsilon$.

Proof. Let $\mathbf{x} \in [0,1]^{d_0}$ be any input vector and let $f_\mathbf{x} \in \mathcal{CS}(L,d_0,d_L)$ be its induced input signal given the parameters L, d_0, and d_L. Additionally, let $K \in \mathcal{CK}(B,L,d_0,d_L)$ denote the computational kernel of Θ.

Let $U^{(L)}$ be the output layer of K (of length $1/(L+2)$) and $\mathcal{C}_{d_L}^{\text{out}}$ be the coarse output partition of K, which partitions the interval $U^{(L)}$ into d_L intervals of length $1/(L+2)d_L$, i.e. the Lebesgue measure μ of each of the intervals is $1/(L+2)d_L$.

By Theorem 4.2, for any $\epsilon > 0$, there exists a computational kernel K' with hidden dimension

$$d = 8ML\left\lceil \frac{2^{2^{\left\lceil \frac{16(L+2)^2 d_L^2(2B)^{2L}}{\epsilon^2}\right\rceil}}(L+2)d_L(2B)^L}{\epsilon}\right\rceil \quad (8)$$

approximating K in cut norm with error

$$\delta_\square^{\text{comp}}(K,K') < \frac{\epsilon}{(L+2)d_L(2B)^L}.$$

By Remark 3.1 (for any computational kernel there is a dense neural network that induces it) there exists a dense network $\Theta' \in \mathcal{NN}_{\text{dense}}(B,L,d_0,d_L)$, with hidden dimension given by (8), that induces K'.

Let $j \in [d_L]$ be any output channel and denote the jth interval of $\mathcal{C}_{d_L}^{\text{out}}$ by I_j^{out}. By Lemma 3.2, we have

$$\Theta(\mathbf{x})_j = \Phi_{B,L}\big(K, f_\mathbf{x}\big)\big(v\big),$$
$$\Theta'(\mathbf{x})_j = \Phi_{B,L}\big(K', f_\mathbf{x}\big)\big(v\big)$$

whenever $v \in I_j^{\text{out}}$. Thus,

$$\begin{aligned}&\big|\Theta(\mathbf{x})_j - \Theta'(\mathbf{x})_j\big| \\ &\qquad = \big|\Phi_{B,L}(K,f_\mathbf{x})(v) - \Phi_{B,L}(K',f_\mathbf{x})(v)\big|\end{aligned}$$

whenever $v \in I_j^{\text{out}}$. Therefore, for any $v \in I_j^{\text{out}}$:

$$\begin{aligned}&\big|\Phi_{B,L}(K,f_\mathbf{x})(v) - \Phi_{B,L}(K',f_\mathbf{x})(v)\big| \\ &= \frac{1}{\mu\left(I_j^{\text{out}}\right)} \\ &\qquad \cdot \left|\mu\left(I_j^{\text{out}}\right)\left(\Phi_{B,L}(K,f_\mathbf{x})(v) - \Phi_{B,L}(K',f_\mathbf{x})(v)\right)\right| \\ &= (L+2)d_L \\ &\qquad \cdot \left|\int_{I_j^{\text{out}}} \left(\Phi_{B,L}(K,f_\mathbf{x})(u) - \Phi_{B,L}(K',f_\mathbf{x})(u)\right) du\right| \\ &\le (L+2)d_L \|\text{Out}_{K,f_\mathbf{x}} - \text{Out}_{K',f_\mathbf{x}}\|_\square,\end{aligned}$$

where

$$\text{Out}_{K,f_\mathbf{x}} := \mathbb{1}_{U^{(L)}}\Phi_{B,L}(K,f_\mathbf{x}),$$
$$\text{Out}_{K',f_\mathbf{x}} := \mathbb{1}_{U^{(L)}}\Phi_{B,L}(K',f_\mathbf{x}).$$

Lastly, by the Lipschitz property of the B-IR-MPNN (Theorem 4.1)

$$\begin{aligned}&(L+2)d_L\|\text{Out}_{K,f_\mathbf{x}} - \text{Out}_{K',f_\mathbf{x}}\|_\square \\ &\qquad \le (L+2)d_L(2B)^L\,\delta_\square^{\text{comp}}(K,K') < \epsilon.\end{aligned}$$

Hence,

$$\big|\Theta(\mathbf{x})_j - \Theta'(\mathbf{x})_j\big| < \epsilon.$$

□

C. Implications for Approximation

In this section, we prove our main results, providing explicit conditions under which dense deep ReLU networks fail to be universal. In other words, for sufficiently large

input dimension d_0, *any dense ReLU network architecture is not dense in* $\mathrm{Lip}(d_0, d_L)$ (the space of Lipschitz continuous functions with Lipschitz constant bounded by 1). First, we prove a variant of (Yarotsky, 2017), Theorem 4., part (a), which depends only on one implicit parameter (in contrast to two in the original theorem). (Yarotsky, 2017), Theorem 4. provides lower bounds on the number of computational units a network architecture must have in order to be able to approximate any function in $\mathrm{Lip}(d_0, d_L)$. Then, we use these two results to show that for a large enough input dimension, no ReLU architecture can approximate all functions with a small ϵ error. This highlights an inherent limitation of dense networks.

C.1. Lower Bounds Based on VC-Dimension

We establish the following theorem by following the steps of the proof of (Yarotsky, 2017), Theorem 4. Theorem 5.1 gives a result close to the first part of (Yarotsky, 2017), Theorem 4, but with explicit constants restricted to Lipschitz continuous functions.

Recall that for a hypothesis class $\mathcal{H}$ of Boolean functions on $[0,1]^d$, the *VC-dimension* is the size of the largest subset $S \subset [0,1]^d$ that can be shattered by $\mathcal{H}$, i.e., on which $\mathcal{H}$ can realize all dichotomies (see Section 2).

Of particular interest is the case where $\mathcal{H}$ consists of Boolean functions obtained by thresholding the scalar output of a ReLU network with fixed architecture and variable weights and biases. Concretely, let $\Theta_{(\mathbf{W},\mathbf{b})} : [0,1]^d \to \mathbb{R}$ denote the function computed by such a network with parameters $(\mathbf{W}, \mathbf{b})$, and consider the class

$$\mathcal{H} = \big\{ x \mapsto \mathbb{1}_{\{f_\theta(x) > a\}} \;\big|\; (\mathbf{W}, \mathbf{b}) \text{ network parameters }, \; a \in \mathbb{R} \big\}.$$

Here, the parameter $a \in \mathbb{R}$ represents a threshold applied to the network output. When the network parameters are unconstrained, this threshold can be fixed to $a = 0$ without loss of generality, since it can be absorbed into the bias of the last layer. We emphasize that no margin assumption is imposed here; the thresholding is purely combinatorial and used only to define a Boolean-valued hypothesis class.

In this setting, (Anthony and Bartlett, 2009), Theorem 8.7 shows that the VC-dimension satisfies

$$\mathrm{VCdim}(\mathcal{H}) \le c \cdot W^2, \tag{9}$$

where W is the total number of weights in the network and c is a universal constant independent of the architecture. Recall that $\mathrm{Lip}(d_0, d_L)$ denotes the space of 1-Lipschitz continuous functions from $[0,1]^{d_0}$ to $\mathbb{R}^{d_L}$ bounded by 1. Theorem 7 of (Harvey et al., 2017) presents an explicit bound that improves upon the classical VC-dimension bound in Equation (9), showing that the universal constant c is in fact small.

Theorem 5.1. *For any $\epsilon \in (0, 1/3)$, a ReLU network architecture capable of approximating every function $f \in \mathrm{Lip}(d_0, d_L)$ with error bounded by ϵ must have at least $c^{-1/2}(6\epsilon)^{-d_0/2}$ parameters, where c is the absolute constant defined above.*

Proof. We first reduce to the case of a scalar output. Since the Lipschitz constant and the approximation error are defined using the infinity norm over the output, every output channel can be treated independently. Hence, it suffices to prove the result for $d_L = 1$.

Given a positive integer N to be chosen later, choose S as a set of N^{d_0} points $\mathbf{x}_1, \ldots, \mathbf{x}_{N^{d_0}}$ in the cube $[0,1]^{d_0}$ such that the Euclidean distance between any pair of them is not less than $\frac{1}{N}$. Given any assignment of values $y_1, \ldots, y_{N^{d_0}} \in \mathbb{R}$, we can construct a Lipschitz continuous function f satisfying $f(\mathbf{x}_m) = y_m$ for all m by setting

$$f(\mathbf{x}) = \sum_{m=1}^{N^{d_0}} y_m \phi\big(N(\mathbf{x} - \mathbf{x}_m)\big), \tag{10}$$

with $\phi : \mathbb{R}^{d_0} \to \mathbb{R}$ such that $\phi(\mathbf{x}) = 1 - 2\|\mathbf{x}\|_2$ if $0 \le \|x\|_2 < \frac{1}{2}$ and $\phi(\mathbf{x}) = 0$ if $\|\mathbf{x}\| \ge \frac{1}{2}$. Notice that f is continuous and differentiable almost everywhere. Let us obtain a condition ensuring that such $f \in \mathrm{Lip}(d_0, d_L)$. It is easy to see that

$$\max_{\mathbf{x}} |f(\mathbf{x})| \le \max_m |y_m|$$
$$\text{and}$$
$$\max_{\mathbf{x}} |\nabla f(\mathbf{x})| = N \max_m |y_m| \max_{\mathbf{x}} |\nabla \phi(\mathbf{x})|.$$

Thus, we obtain

$$\max_x |\nabla f(\mathbf{x})| \le 2N \max_m |y_m|. \tag{11}$$

Therefore, if $\max_m |y_m| \le \frac{1}{2N}$, then is 1-Lipschitz. Now, set

$$N = \frac{1}{6\epsilon}. \tag{12}$$

Then $\max_{\mathbf{x}} |f(\mathbf{x})| \le 3\epsilon$. In particular, when $\epsilon \le 1/3$, $\max_{\mathbf{x}} |f(\mathbf{x})| \le 1$. Therefore, $f \in \mathrm{Lip}(d_0, d_L)$. Suppose that there is a ReLU network architecture η that can approximate, by adjusting its weights, any $f \in \mathrm{Lip}(d_0, d_L)$ with error less than ϵ. By $\Theta_{(\mathbf{W},\mathbf{b})}(\mathbf{x})$, we denote the output of the network with parameters $(\mathbf{W}, \mathbf{b})$.

Consider any assignment $\mathbf{z}$ of Boolean values $z_1, \ldots, z_{N^{d_0}} \in \{0, 1\}$. Set

$$y_m = \frac{z_m}{2N}, \quad m = 1, \ldots, N^{d_0},$$

and let f be given by Equation (10); then Equation (11) holds and hence $f \in \mathrm{Lip}(d_0, d_L)$. By assumption, there

exist fixed parameters $(\mathbf{W}_z, \mathbf{b}_z)$, such that for all m we have

$$|\Theta_{(\mathbf{W}_z,\mathbf{b}_z)}(\mathbf{x}_m) - y_m| \leq \epsilon,$$

and in particular

$$\Theta_{(\mathbf{W}_z,\mathbf{b}_z)}(\mathbf{x}_m) = \begin{cases} \geq \frac{1}{2N} - \epsilon > \frac{1}{4N}, & \text{if } z_m = 1, \\ \leq \epsilon < \frac{1}{4N}, & \text{if } z_m = 0, \end{cases}$$

so the thresholded network $\hat{\Theta}_{(\mathbf{W}_z,\mathbf{b}_z)} = \mathbf{1}(\Theta_{(\mathbf{W}_z,\mathbf{b}_z)} > \frac{1}{4N})$ has outputs

$$\hat{\Theta}_{(\mathbf{W}_z,\mathbf{b}_z)}(\mathbf{x}_m) = z_m, \quad m = 1, \ldots, N^{d_0}.$$

Since the Boolean values z_m were arbitrary, we conclude that the subset S is shattered and hence

$$\mathrm{VCdim}(\hat{\Theta}) \geq N^{d_0}.$$

Expressing N through ϵ with Equation (12), we obtain

$$\mathrm{VCdim}(\hat{\Theta}) \geq (6\epsilon)^{-d_0}. \tag{13}$$

To establish the Theorem, we apply the inequality in Equation (9) to the network $\hat{\Theta}$:

$$\mathrm{VCdim}(\hat{\Theta}) \leq cW^2, \tag{14}$$

where W is the number of weights in η_1, which is the same as in η if we do not count the threshold parameter. Combining Equation (13) with Equation (14), we obtain the desired lower bound

$$W \geq c^{-1/2}\,(6\epsilon)^{-d_0/2}.$$

□

Next, we relate the number of parameters of a neural network in $\mathcal{NN}_{\text{dense}}(B, L, d_0, d_L)$ to the hidden and input dimensions, using formulae compatible with Corollary 4.3 and Theorem 5.1 with the choice of $\epsilon = 1/8$. This correspondence will be used to derive the expressivity bound on strongly dense neural networks.

Lemma C.1. *Let $\epsilon_0 = 1/8$. Let $B \geq 1$, $L, d_L \in \mathbb{N}$, and let c be the constant from Theorem 5.1. Let d_0 satisfy*

$$\begin{aligned} d_0 \geq & 17 \log_2\big(c^{1/2} L^3 (L+2)^2 d_L^4 (2B)^{2L}\big) \\ & + 17 \cdot 2^{14} (L+2)^2 d_L^2 (2B)^{2L} + 306. \end{aligned}$$

Consider the family $\mathcal{H}_{\epsilon_0}$ of ReLU networks with strictly less than $c^{-1/2}(6\epsilon_0)^{-d_0/2}$ parameters. Then, the space of dense neural networks in $\mathcal{NN}_{\text{dense}}(B, L, d_0, d_L)$ with hidden dimension

$$d = 8ML \left\lceil \frac{2^{2^{\left\lceil \frac{16(L+2)^2 d_L^2 (2B)^{2L}}{(\epsilon_0/2)^2} \right\rceil}} (L+2) d_L (2B)^L}{(\epsilon_0/2)} \right\rceil$$

is a subset of $\mathcal{H}_{\epsilon_0}$.

Proof. We now show that any network $\Theta \in \mathcal{NN}_{\text{dense}}(B, L, d_0, d_L)$ with hidden dimension

$$d = 8ML \left\lceil \frac{2^{2^{\left\lceil \frac{16(L+2)^2 d_L^2 (2B)^{2L}}{(\epsilon_0/2)^2} \right\rceil}} (L+2) d_L (2B)^L}{(\epsilon_0/2)} \right\rceil$$

has fewer than $c^{-1/2}(6\epsilon_0)^{-d_0/2}$ parameters. By substituting $\epsilon_0 = 1/8$ and collecting powers of 2, we obtain

$$\begin{aligned} & \left\lceil \frac{2^{2^{\left\lceil \frac{16(L+2)^2 d_L^2 (2B)^{2L}}{(\epsilon_0/2)^2} \right\rceil}} (L+2) d_L (2B)^L}{(\epsilon_0/2)} \right\rceil \\ & = \left\lceil 2^{2^{\lceil 2^{12}(L+2)^2 d_L^2 (2B)^{2L} \rceil + 4}} (L+2) d_L (2B)^L \right\rceil \end{aligned}$$

Notice that

$$\begin{aligned} & \left\lceil 2^{2^{\lceil 2^{12}(L+2)^2 d_L^2 (2B)^{2L} \rceil + 4}} (L+2) d_L (2B)^L \right\rceil \\ & \leq \left(2^{2^{(2^{12}(L+2)^2 d_L^2 (2B)^{2L} + 1) + 4}} (L+2) d_L (2B)^L + 1 \right) \\ & = \left(2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L} + 6} (L+2) d_L (2B)^L + 1 \right) \end{aligned}$$

Therefore,

$$\begin{aligned} d \leq & 8ML \cdot 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L} + 6} (L+2) d_L (2B)^L \\ & + 8ML \\ = & 2^3 ML \cdot 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L} + 6} (L+2) d_L (2B)^L \\ & + 2^3 ML \\ = & (ML(L+2) d_L (2B)^L) 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L} + 9} \\ & + 2^3 ML. \end{aligned}$$

Since $M \leq d_0 d_L$, we have

$$\begin{aligned} d \leq & (d_0 d_L^2 L (L+2)(2B)^L) 2^{2^{13}(L+2)^2 (2B)^{2L} + 9} \\ & + 2^3 d_0 d_L L. \end{aligned}$$

We now calculate W, the number of parameters (including both the weights and the biases) of Θ. We have

$$W = (L-1)d^2 + (d_0 + d_L + L - 1)d + d_L,$$

since there are $(L-1)d^2 + (d_0 + d_L)d$ weights and $(L-1)d + d_L$ biases in a fixed-width network.

Since $d \geq d_L$, we have

$$(d_0 + d_L + L - 1)d + d_L \leq (d_0 + d_L + L)d.$$

Since $L \geq 2$, we have

$$(d_0 + d_L + L)d \leq d_0 d_L L d.$$

Thus

$$W = (L-1)d^2 + (d_0 + d_L + L - 1)d + d_L \leq (L-1)d^2 + d_0 d_L L d.$$

The bound

$$d \leq (d_0 d_L^2 L(L+2)(2B)^L) 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L}+9} + 2^3 d_0 d_L L$$

yields

$$\begin{aligned}
W \leq & (L-1) \cdot \Big(\big(d_0 d_L^2 L(L+2)(2B)^L\big) \cdot 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L}+9} + 2^3 d_0 d_L L \Big)^2 \\
& + d_0 d_L L \cdot \Big(\big(d_0 d_L^2 L(L+2)(2B)^L\big) 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L}+9} + 2^3 d_0 d_L L \Big) \\
= & (L-1) \cdot \big(d_0 d_L^2 L(L+2)(2B)^L\big)^2 \cdot \Big(2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L}+9}\Big)^2 \\
& + (L-1) 2^3 d_0 d_L L \cdot \big(d_0 d_L^2 L(L+2)(2B)^L\big) \cdot 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L}+9} \\
& + (L-1)\left(2^3 d_0 d_L L\right)^2 \\
& + d_0 d_L L \cdot \Big((d_0 d_L^2 L(L+2)(2B)^L) \cdot 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L}+9} \Big) \\
& + d_0 d_L L \cdot \big(2^3 d_0 d_L L\big) \\
= & (L-1) d_0^2 d_L^4 L^2 (L+2)^2 (2B)^{2L} \cdot 2^{2^{14}(L+2)^2 d_L^2 (2B)^{2L}+18} \\
& + (L-1) d_0^2 d_L^3 L^2 (L+2)(2B)^L \cdot 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L}+12} \\
& + (L-1) 2^6 d_0^2 d_L^2 L^2 \\
& + d_0^2 d_L^3 L^2 (L+2)(2B)^L \cdot 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L}+9} \\
& + 2^3 d_0^2 d_L^3 L^2 \\
=: & (*)
\end{aligned}$$

Notice that

$$\begin{aligned}
& (L-1) 2^6 d_0^2 d_L^2 L^2 + d_0^2 d_L^3 L^2 (L+2)(2B)^L 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L}+9} + 2^3 d_0^2 d_L^3 L^2 \\
& \leq 2 \cdot d_0^2 d_L^3 L^2 (L+2) d_L (2B)^L 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L}+9} \\
& = d_0^2 d_L^3 L^2 (L+2) d_L (2B)^L 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L}+10} \\
& \leq d_0^2 d_L^3 L^2 (L+2) d_L (2B)^L 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L}+12}
\end{aligned}$$

Thus

$$\begin{aligned}
& (L-1) d_0^2 d_L^3 L^2 (L+2)(2B)^L 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L}+12} \\
& \quad + (L-1) 2^6 d_0^2 d_L^2 L^2 \\
& \quad + d_0^2 d_L^3 L^2 (L+2)(2B)^L 2^{2^{13}(L+2)^2 d_L^2 (2B)^{2L}+9} \\
& \quad + 2^3 d_0^2 d_L^3 L^2 \\
& \leq (L+2) M d_0^2 d_L^4 L^2 (L+2)(2B)^{2L} \cdot 2^{2^{14}(L+2)^2 d_L^2 (2B)^{2L}+18} \\
& = d_0^2 d_L^4 L^2 (L+2)^2 (2B)^{2L} 2^{2^{14}(L+2)^2 d_L^2 (2B)^{2L}+18}
\end{aligned}$$

All in all, we get

$$\begin{aligned}
(*) & \leq d_0^2 d_L^4 L^3 (L+2)^2 (2B)^{2L} 2^{2^{14}(L+2)^2 d_L^2 (2B)^{2L}+18} \\
& \leq d_0^2 d_L^4 L^3 (L+2)^2 (2B)^{2L} 2^{2^{14}(L+2)^2 d_L^2 (2B)^{2L}+18} \\
& := \widetilde{W}
\end{aligned}$$

That is, $W \leq \widetilde{W}$. We seek a condition on d_0 such that

$$\widetilde{W} \leq c^{-1/2} \left(\frac{8}{6}\right)^{d_0/2},$$

which implies $W \leq c^{-1/2}(6\epsilon_0)^{-d_0/2}$. Taking logarithms, this condition is equivalent to

$$\log\big(c^{1/2}\widetilde{W}\big) \leq \frac{d_0}{2} \log\left(\frac{4}{3}\right).$$

Substituting the expression for $\widetilde{W}$ yields

$$\begin{aligned}
\log_2\big(c^{1/2} L^3 (L+2)^2 d_L^4 (2B)^{2L}\big) & + 2^{14}(L+2)^2 d_L^2 (2B)^{2L} + 18 \leq \frac{d_0}{2} \log\left(\frac{4}{3}\right) - 2\log(d_0).
\end{aligned}$$

A direct calculation shows that for all $d_0 > 1800$,

$$\frac{1}{17} d_0 \leq \frac{d_0}{2} \log\left(\frac{4}{3}\right) - 2\log(d_0).$$

Therefore, it suffices to require

$$17\log_2\big(c^{1/2}L^3(L+2)^2d_L^4(2B)^{2L}\big) + 17\cdot 2^{14}(L+2)^2d_L^2(2B)^{2L} + 306 \le d_0,$$

which is precisely the equation we wanted. □

From the above theorem, we get the following.

Theorem 5.2 (Expressivity Bound of Strongly Dense Neural Networks)**.** *Let $L, d_L \in \mathbb{N}$ with $L \ge 2$, let $B \ge L+2$, and let c be the constant from Theorem 5.1. Let d_0 satisfy*

$$d_0 \ge 17\log_2\big(c^{1/2}L^3(L+2)^2d_L^4(2B)^{2L}\big) + 17\cdot 2^{14}(L+2)^2d_L^2(2B)^{2L} + 306.$$

Then $\mathcal{NN}_{\text{dense}}(B, L, d_0, d_L)$ is not a universal approximator of $\text{Lip}(d_0, d_L)$.

Proof. Fix $\epsilon_0 := 1/8$. Consider the family $\mathcal{H}_{\epsilon_0}$ of ReLU networks with at most $c^{-1/2}(6\epsilon_0)^{-d_0/2}$ parameters. By Theorem 5.1, there exists $f \in \text{Lip}(d_0, d_L)$ such that no function from $\mathcal{H}_{\epsilon_0}$ can approximate it with error less than ϵ_0. On the other hand, by Lemma C.1 the space of dense neural networks $\mathcal{NN}_{\text{dense}}(B, L, d_0, d_L)$ with hidden dimension

$$d = 8ML\left\lceil \frac{2^{2\left\lceil \frac{16(L+2)^2d_L^2(2B)^{2L}}{\epsilon_0/2)^2}\right\rceil}(L+2)d_L(2B)^L}{(\epsilon_0/2)} \right\rceil$$

is a subset of $\mathcal{H}_{\epsilon_0}$.

Suppose, for the sake of contradiction, that $\mathcal{NN}_{\text{dense}}(B, L, d_0, d_L)$ is a universal approximator of $\text{Lip}(d_0, d_L)$. Then there exists $\Theta \in \mathcal{NN}_{\text{dense}}(B, L, d_0, d_L)$ such that

$$\|f - \Theta\|_\infty \le \epsilon_0/2.$$

By Corollary 4.3, any dense ReLU network $\Theta \in \mathcal{NN}_{\text{dense}}(B, L, d_0, d_L)$ admits an equivalent dense ReLU network $\Theta' \in \mathcal{NN}_{\text{dense}}(B, L, d_0, d_L)$ with controlled hidden dimension

$$d = 8ML\left\lceil \frac{2^{2\lceil (16(L+2)^2d_L^2(2B)^{2L})/\epsilon_0^2\rceil}(L+2)d_L(2B)^L}{\epsilon_0} \right\rceil$$

such that

$$\|\Theta - \Theta'\|_\infty \le \epsilon_0/2.$$

By the triangle inequality,

$$\|f - \Theta'\|_\infty \le \|f - \Theta\|_\infty + \|\Theta - \Theta'\|_\infty \le \epsilon_0.$$

This contradiction completes the proof. □

D. Experiments

We empirically test the theoretical prediction that increasing width in dense networks leads to early saturation in performance. We use fully connected one-hidden-layer ReLU networks in PyTorch, trained on MNIST with the Adam optimizer (learning rate 10^{-3}, batch size 128, 20 epochs). We consider two training modes: A *Standard* mode, where networks are trained without constraints and a *Dense* mode, where weights are clamped after each optimizer step to the interval $[-10/d_{i-1},\ 10/d_{i-1}]$, where d_{i-1} is the input dimension of the layer. We vary the hidden-layer width from small to highly overparameterized regimes, recording final training and test accuracy for each setting. Experiments were run on a single NVIDIA GeForce RTX 3050 6GB GPU. The code we used in this paper will be available upon the acceptance of this paper.

Results. Standard networks improve rapidly in the underparameterized regime and reach almost perfect performance, with training accuracy stabilizing around 98–99%. Dense networks plateau earlier, with training accuracy around 90–91% (Table 1), and achieve lower test accuracy across all widths (Table 2), creating a persistent performance gap. This supports the theoretical prediction that width beyond a certain scale does not further increase the effective expressive power of dense networks.

Width	Standard Train Acc (%)	Dense Train Acc (%)
1	29.5 ± 6.8	27.8 ± 6.2
2	50.5 ± 11.6	43.3 ± 12.1
4	81.0 ± 2.9	77.0 ± 4.8
8	90.0 ± 1.4	86.5 ± 1.0
16	94.5 ± 1.9	89.2 ± 0.2
32	96.9 ± 2.6	89.9 ± 0.4
64	98.2 ± 3.1	90.5 ± 0.5
128	98.9 ± 2.8	90.8 ± 0.3
256	98.9 ± 2.8	90.6 ± 0.4
512	98.9 ± 3.0	90.8 ± 0.3
1024	98.9 ± 2.9	90.9 ± 0.4
2048	98.7 ± 3.1	90.8 ± 0.4

Table 1. Training accuracy on MNIST for standard and dense one-hidden-layer ReLU networks as a function of width. Dense networks enforce per-layer weight constraints $[-10/d_{i-1},\ 10/d_{i-1}]$. Values denote mean $\pm$ standard deviation over random seeds.

E. Deep Neural Networks as Graph-Based Models

In this section, we define the computational graph of a feedforward network and formalize, in Lemma E.1, the connection between graph-based neural computations and ReLU networks. We then present Lemma E.2, which is a direct consequence of the analysis in Appendix E of (Levie, 2023) and (Rauchwerger and Levie, 2025), con-

Width	Standard Test Acc (%)	Dense Test Acc (%)
1	29.5 ± 6.8	27.9 ± 6.2
2	50.5 ± 11.5	43.6 ± 11.7
4	80.7 ± 3.1	77.5 ± 5.1
8	89.5 ± 1.1	86.8 ± 0.8
16	93.8 ± 1.6	89.6 ± 0.2
32	95.8 ± 2.1	90.2 ± 0.4
64	96.8 ± 2.5	90.8 ± 0.5
128	97.3 ± 2.1	91.1 ± 0.4
256	97.5 ± 2.2	90.8 ± 0.4
512	97.5 ± 2.5	91.1 ± 0.4
1024	97.5 ± 2.2	91.2 ± 0.4
2048	97.3 ± 2.5	91.0 ± 0.4

Table 2. Test accuracy on MNIST for standard and dense one-hidden-layer ReLU networks as a function of width. Dense networks enforce per-layer weight constraints $[-10/d_{i-1},\ 10/d_{i-1}]$. Values denote mean $\pm$ standard deviation over random seeds.

necting graph-based neural computations to kernel-based neural computations.

Graph Neural Networks and specifically *message passing networks (MPNNs)*, form a class of neural networks designed to process graph-structured data (Xu et al., 2019; Rossi et al., 2023) by iteratively updating node embeddings through the exchange of messages between nodes. Unlike general trainable MPNNs, we consider the following special predefined network.

SR-MPNN. Let $L \in \mathbb{N}$; the L-layer *sum-ReLU MPNN (SR-MPNN)* processes attributed graphs into output vertex attributes as follows. The SR-MPNN is applied on an attributed graph $(G, \mathbf{f})$ with adjacency matrix $\mathbf{A} = (A_{i,j})_{i,j\in[n]}$ as the function defined by $\Psi_L(G, \mathbf{f}) := \mathbf{f}^{(L)}$, where $\mathbf{f}^{(0)} := \mathbf{f}$ and for each layer $\ell = 1, \ldots, L-1$ the update rule is:

$$\mathbf{f}^{(\ell)}(i) := \mathrm{ReLU}\left(\frac{1}{n}\sum_{j\in[n]} A_{i,j}\mathbf{f}^{(\ell-1)}(j)\right). \qquad (15)$$

The output is computed like in Equation (15) with $\ell = L$, without ReLU.

The computational graph of a neural network. To represent a feedforward neural network within the SR-MPNN framework, we associate it with a directed computational graph in which vertices may be repeated. These repetitions are a technical but essential feature of the construction.

Intuitively, when a feedforward network is implemented as an SR-MPNN on its computational graph, message passing uses *normalized sum aggregation*. If each neuron were represented by a single vertex, this normalization would excessively downscale signals as they propagate through the layers, so that the resulting SR-MPNN would no longer implements the intended feedforward computation. To balance the contributions of different neurons and ensure that signal magnitudes are preserved across layers, we therefore repeat each input, bias, and output neuron several times; the precise reason for this repetition will become clear later in the analysis.

A *partition* of the vertex set V of a graph is a sequence of subsets $(V_j \subset V)_{j=1}^{J}$ such that $\cup_j V_j = V$ and $V_j \cap V_i = \emptyset$ for every $i \neq j$. The partition is called *balanced* if $|V_i| = |V_j|$ for every i, j, where $|V_i|$ is the number of vertices in V_i.

We call a graph $G = (V, E)$, with an adjacency matrix $\mathbf{A} = (A_{v,u})_{v,u\in[n]}$, a depth-$L$ *computational graph* with respect to the parameters $L, d_0, d_L \in \mathbb{N}$, if G satisfies the following four conditions.

- *Condition 1.* The graph is a weighted graph. The size of the graph n is divisible by $M(L+2)$, where M is the least common multiple of d_0 and d_L.

  Notation: The vertex set V is endowed with a balanced partition into $L+2$ sets of size $d := n/(L+2)$:

  $$V = V^{(0)} \cup V^{(1)} \cup \cdots \cup V^{(L)} \cup V^{((\mathrm{bias}))},$$

  where each $V^{(\ell)}$, $\ell = 0, \ldots, L$, is called *layers*, and $V^{((\mathrm{bias}))}$ is called the *bias*. Specifically, we call $V^{(0)}$ the *input layer* and $V^{(L)}$ the *output layer*. The layers $V^{(0)}$ and $V^{(L)}$ are endowed with balanced partitions $(V_i^{(\mathrm{in})})_{i=1}^{d_0}$ and $(V_i^{(\mathrm{out})})_{i=1}^{d_L}$ into d_0 and d_L cells, respectively, each part having size d/d_0 and d/d_L. We call $(V_i^{(\mathrm{in})})_{i=1}^{d_0}$ *input partition* and $(V_i^{(\mathrm{out})})_{i=1}^{d_L}$ the *output partition*.

- *Condition 2.* The graph satisfies $A_{v,u} = A_{v,u'}$ whenever u and u' belong to the same subset in $(V_i^{(\mathrm{in})})_{i=1}^{d_0}$. Moreover, $A_{v,u} = A_{v',u}$ whenever v and v' belong to the same interval of $(V_i^{(\mathrm{out})})_{i=1}^{d_L}$. In addition, $A_{v,u} = A_{v,u'}$ whenever u and u' belong to $V^{(\mathrm{bias})}$.

- *Condition 3.* The graph satisfies $A_{v,u} = 0$ whenever either $u \in V^{(\ell)}$ and $v \notin V^{(\ell+1)}$ for some $\ell \in \{0, \ldots, L-1\}$, or $u \in V^{(\mathrm{bias})}$ and $v \in V^{(0)}$, or $v \in V^{(\mathrm{bias})}$ and $u \notin V^{(\mathrm{bias})}$.

- *Condition 4.* For any $v, u \in V^{(\mathrm{bias})}$, $A_{v,u} = (L+2)$.

We call d_0 the *input dimension*, d_L the *output dimension*, and refer to d as the *hidden dimension* for $\ell = 1, \ldots, L-1$. By $\mathcal{CG}(L, d_0, d_L)$, we denote the collection of all depth-L computational graphs as defined above.

Computational Node Feature Vectors. Given parameters L, d_0, d_L, a *computational node feature vector* is any

node feature vector which is constant on each part in $C^{\text{in}}_{d_0}$ and $C^{\text{out}}_{d_L}$, and constant on $V^{(\text{bias})}$. We define a *computational input node feature vector* as any computational node feature vector satisfying $\mathbf{f}(v) = 0$ for all $v \notin V^{(0)} \cup V^{(\text{bias})}$. The values of $\mathbf{f}$ on the sets of $C^{\text{in}}_{d_0}$ are interpreted as the input vector to the network, while the constant value on $V^{(\text{bias})}$ represents a bias signal that is present at every layer. *Condition 4* of computational graph, i.e., $A_{v,u} = (L+2)$ for all $v, u \in V^{(\text{bias})}$, ensures that this bias signal is unchanged during propagation, i.e., under successive applications of the computational kernel on the signal.

Computational Graph Induced by Network Parameters. Given a neural network, one can induce a computational graph on which the SR-MPNN implements the forward propagation of the neural network. Let $(\mathbf{W}, \mathbf{b})$ be the parameters of the neural network $\Theta_{(\mathbf{W},\mathbf{b})} \in \mathcal{NN}(L, d_0, d_L)$, with hidden dimension d divisible by d_0 and d_L. Denote $n = d(L+2)$. We define the computational graph $G = G_{(\mathbf{W},\mathbf{b})} \in \mathcal{CG}(L, d_0, d_L)$ *induced* by $\Theta_{(\mathbf{W},\mathbf{b})}$ as follows. For every $(v, u) \in [n]^2$:

- $A_{v,u} = W^{(1)}_{i,j}$ if u is in $V^{(\text{in})}_j$ and v is the ith vertex of $V^{(1)}$.
- $A_{v,u} = W^{(L)}_{i,j}$ if u is the jth vertex of $V^{(L-1)}$ and v is in $V^{(\text{out})}_i$.
- For $\ell = 2, \dots, L-1$, $K(x, y) = W^{(\ell)}_{i,j}$ if y is the jth vertex of $V^{(\ell-1)}$ and x is the ith vertex of $V^{(\ell)}$.
- For $\ell = 2, \dots, L-1$, $A_{v,u} = b^{(\ell)}_i$ if u is in $V^{(\text{bias})}$ and v is the ith interval of $V^{(\ell)}$. Moreover, $K(x, y) = b^{(L)}_i$ if y is in $U^{(\text{bias})}$ and x is in $V^{(\text{out})}_i$.

We encode an input vector $\mathbf{x} \in \mathbb{R}^{d_0}$ as node attributes $\mathbf{f}_{\mathbf{x}} : V \to \mathbb{R}$ defined by

$$\mathbf{f}_{\mathbf{x}}(v) := \begin{cases} x_i, & \text{if } v \text{ is in } V^{(\text{in})}_i, \\ 1, & \text{if } v \in V_{(\text{bias})}, \\ 0, & \text{otherwise.} \end{cases}$$

In the above construction, when writing "the jth cell of the partition..." we implicitly assume that the cells are always sorted in increasing order.

Networks as SR-MPNNs. Lemma E.1 shows that the output of a B-strongly dense neural network can be expressed in terms of the L-layer SR-MPNN applied to its induced computational kernel.

Lemma E.1. *Let $(\mathbf{W}, \mathbf{b})$ be the parameters of $\Theta_{(\mathbf{W},\mathbf{b})} \in \mathcal{NN}(L, d_0, d_L)$. Then, for any input $\mathbf{x} \in \mathbb{R}^{d_0}$ and output channel $i \in [d_L]$, we have*

$$\Theta_{(\mathbf{W},\mathbf{b})}(\mathbf{x})_i = \Psi_L\big(G_{(\mathbf{W},\mathbf{b})}, f_{\mathbf{x}}\big)(v),$$

whenever v is in $V^{(\text{out})}_i$.

The proof of Lemma E.1 follows the same steps as the proof of Lemma 3.2.

Computational Kernel as An Induced Computational Graph We call any depth-L computational graph with respect to the parameters $L, d_0, d_L \in \mathbb{N}$ a *dense computational graph with bound B* if the graph is $[-B, B]$-weighted for some $B \geq L+2$. In this case, we can induce the computational graph into a computational kernel using the method introduced in Section 2.

(Levie, 2023; Rauchwerger and Levie, 2025) show that applying an MPNN to an attributed graph and then inducing an attributed kernel yields the same representation as first inducing the attributed kernel and then applying the MPNN (see e.g. Appendix E in (Rauchwerger and Levie, 2025)). A direct result of this theorem in our setting is stated as follows.

Lemma E.2. *Let $L \in \mathbb{N}$ and $B \in \mathbb{R}$. For any $[-B, B]$-weighted attributed graph $(G, \mathbf{f})$ with node set $[n]$, applying an L-layer B-IR-MPNN to its induced attributed kernel $(K_G, f_{\mathbf{f}})$ corresponds to applying the standard L-layer ReLU graph network to G in the following way: for every $i \in [n]$ and every x in the interval I_i:*

$$\Phi(G, \mathbf{f})(i) = \Phi_{L,B}(K_G, f_{\mathbf{f}})(x).$$