ARBITRARY-DEPTH UNIVERSAL APPROXIMATION THEOREMS FOR OPERATOR NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

The standard Universal Approximation Theorem for operator neural networks (NNs) holds for arbitrary width and bounded depth. Here, we prove that operator NNs of bounded width and arbitrary depth are universal approximators for continuous nonlinear operators. In our main result, we prove that for non-polynomial activation functions that are continuously differentiable at a point with a nonzero derivative, one can construct an operator NN of width five, whose inputs are real numbers with finite decimal representations, that is arbitrarily close to any given continuous nonlinear operator. We derive an analogous result for non-affine polynomial activation functions. We also show that depth has theoretical advantages by constructing operator ReLU NNs of depth $2k^3 + 8$ and constant width that cannot be well-approximated by any operator ReLU NN of depth k, unless its width is exponential in k.

1 INTRODUCTION

In the approximation theory of neural networks (NNs), universal approximation theorems (UATs) are statements that establish the density of a class of NNs within a space of mappings. Thus, UATs imply that NNs represent a wide variety of mappings when given appropriate weights and biases. A NN is characterized by its activation function (e.g., ReLU, sigmoid), connectivity (e.g., feedforward, recurrent), width (number of neurons per layer), and depth (number of layers). Operator NNs are a family of NNs for approximating nonlinear operators (Chen & Chen, 1995; Kovachki et al., 2021; Lu et al., 2021). These are critical for learning dynamical systems using DeepONets (Lu et al., 2019; Cai et al., 2021; Lanthaler et al., 2021), inverse mapping problems (Adler & Öktem, 2017), and functional data analysis (Rossi et al., 2005). UATs for operator NNs are a fundamental theoretical underpinning for such applications. While there are UATs for wide, shallow operator NNs (Chen & Chen, 1995), we derive the first set of UATs for their deep, narrow counterparts. These results are key to understanding the expressibility of deep operator NNs.

There are well-established theoretical advantages of deep, narrow NNs over wide, shallow ones in terms of expressibility. In particular, there are 3-layer NNs representing radial functions on $\mathbb{R}^d$ that cannot be approximated by a 2-layer NN to more than a constant accuracy, unless its width is exponential in d, where a NN's depth is the number of hidden layers plus one output layer (Eldan & Shamir, 2016). Moreover, for any $k \in \mathbb{Z}$, there are $\Theta(k^3)$-deep NNs of constant width which, when restricted to the unit cube $[0, 1]^d$, cannot be approximated by a NN with $\mathcal{O}(k)$ depth, unless it has $\Omega(2^k)$ width (Telgarsky, 2016). In Section 2, we construct operators that require exponentially wide ReLU NNs, analogous to Telgarsky (2016). Hence, the improved expressibility of deep, narrow operator NNs over shallow, wide ones is similar to standard NNs. UATs for deep, narrow operator NNs are thus needed to establish their approximation power.

In Section 3, we prove that an operator NN of arbitrary depth and constant width is a universal approximator of nonlinear continuous operators if the activation function is continuously differentiable at a point with nonzero derivative. Our key insight is to use input encoding and reduction of truncated values to decrease the width of the NN to a constant. We thus propagate inputs from one layer to the next with a single neuron. We truncate inputs to a number of digits based on a precision $\varepsilon > 0$ and concatenate the truncated values into one value. We extract each truncated input with a decoder function, which we approximate with an arbitrarily deep NN as described in Kidger &

Lyons (2019). A related approach is used in Shen et al. (2021a;b), where inputs are used in their encoded forms. However, we extract the original value from its encoding.

Our work builds on well-established UATs. An early UAT by Pinkus (1999) states that an arbitrarily wide one-layer NN with a continuous non-polynomial activation function can approximate all continuous functions on compact sets. Kidger & Lyons (2019) prove a UAT for arbitrarily deep NNs with n inputs, m outputs, and of width $n + m + 2$. An arbitrarily wide operator NNs UAT is given in Chen & Chen (1995). They prove the standard UAT with a fixed set of weights and biases, then give an arbitrary-width UAT for nonlinear continuous functionals and operators. Though Kidger & Lyons (2019) turns the arbitrary-width UAT into one of arbitrary depth, their technique does not extend to operator NNs. In particular, it requires the width of the NN to depend on the size of the sampling device, which depends on the precision ε. Consequently, we would have $n+m(\varepsilon)+5$ neurons in every hidden layer of the operator NN. This is impractical, since in most cases $m(\varepsilon) \to \infty$ as $\varepsilon \to 0$, resulting in a NN that is both deep and wide. In contrast, our result only requires a constant width operator NN.

The above UATs all utilize the multi-layer feedforward perceptron (MLP) model. Given an input vector $\mathbf{x} \in \mathbb{R}^{k_0}$ and activation function $\sigma : \mathbb{R} \to \mathbb{R}$, the output $\varphi(\mathbf{x}) \in \mathbb{R}^{k_N}$ is calculated as

$$\varphi(\mathbf{x}) = \mathbf{W}_N \sigma \left(\mathbf{W}_{N-1} \left(\cdots \left(\mathbf{W}_2 \sigma \left(\mathbf{W}_1 \mathbf{x} + \boldsymbol{\theta}_1 \right) + \boldsymbol{\theta}_2 \right) + \cdots \right) + \boldsymbol{\theta}_{N-1} \right) + \boldsymbol{\theta}_N,$$

where $k_0, \ldots, k_N \in \mathbb{N}$, with weights $\mathbf{W}_i \in \mathbb{R}^{k_i \times k_{i-1}}$ and biases $\boldsymbol{\theta}_i \in \mathbb{R}^{k_i}$. Here, σ is applied entry-wise to the vector, i.e., $\sigma(\mathbf{a})_j = \sigma(a_j)$. UATs concern the density of the following space:

$$\mathcal{M}(\sigma) := \text{span}\{\sigma(\mathbf{w}^\top \mathbf{x} - \theta) \mid \theta \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^n\},$$

where n is the input dimension. We say σ has the *density property* if $\mathcal{M}(\sigma)$ is dense in $\mathcal{C}(\mathbb{R}^n)$ equipped with the topology of uniform convergence on compact sets. The definition of the density property is independent of the dimension n of the input space. Moreover, all continuous, non-polynomial activation functions have the density property (Pinkus, 1999).

Main Contributions. We show that deep, narrow NNs are better than shallow, wide ones at approximating certain continuous nonlinear operators, in the sense that significantly fewer neurons are needed to achieve the same accuracy (see Theorem 1). We also show that after truncating inputs, deep NNs of width five can be used to uniformly approximate continuous real-valued functions on compact sets, regardless of the domain's dimension (see Theorems 3 & 4). Finally, we give the first arbitrary-depth UAT for operators with a general class of activation functions (see Theorem 5).

2 Advantages of Depth for Operator Neural Networks

There are many advantages of deep, narrow NNs over wide, shallow ones. In particular, some functions are computable by a NN with two hidden layers but require exponentially many neurons of a NN with one hidden layer (Eldan & Shamir, 2016). This demonstrates the expressive power of deep NNs. However, this result is achieved by considering the L^2 distance between two functions on the entirety of $\mathbb{R}^d$. As our main results are concerned with approximating continuous functions and operators on compact sets, we prove the following more powerful result than Eldan & Shamir (2016) for the operator case, inspired by Theorem 1.1 of Telgarsky (2016).

Theorem 1. *Let $\mathbb{X}$ be a Banach space, $\mathbb{K}_1 \subseteq \mathbb{X}$ be compact, and $\mathbb{V} \subseteq \mathcal{C}(\mathbb{K}_1)$ be compact. Then, for any integers $n, k \geq 1$, there exists a nonlinear continuous operator $G_k : \mathbb{V} \to \mathcal{C}\left([0,1]^n\right)$ such that*

1. *There is a ReLU NN $\varphi : [0,1]^n \to \mathbb{R}$ of depth $2k^3 + 8$ and width in $\Theta(1)$ such that $\varphi(\mathbf{y}) = G_k(u)(\mathbf{y})$, for any $u \in \mathbb{V}$ and $\mathbf{y} \in [0,1]^n$.*
2. *Let $m \geq 1$ be an integer. Let $\psi : [0,1]^{n+m} \to \mathbb{R}$ be a ReLU NN with $n + m$ inputs, depth $\leq k$, and $\leq 2^k$ total nodes. Then for any prescribed $x_1, \ldots, x_m \in \mathbb{K}_1$ and $u \in \mathbb{V}$, we have*

$$\int_{[0,1]^d} \left| G_k(u)(\mathbf{y}) - \psi\left(u(x_1), \ldots, u(x_m), \mathbf{y}\right) \right| \, d\mathbf{y} \geq \frac{1}{64}.$$

Proof. Let $k \geq 1$ and $\varphi : [0,1]^n \to \mathbb{R}$ be the ReLU NN constructed in Theorem 1.1 of Telgarsky (2016) with depth $2k^3 + 8$ and width in $\Theta(1)$. The first statement of our theorem follows from

considering the constant operator $G_k : \mathbb{V} \to \mathcal{C}([0,1]^n)$, $u \mapsto \varphi$. To prove 2, let $\psi : [0,1]^{n+m} \to \mathbb{R}$ be any ReLU NN of depth $\leq k$ with $\leq 2^k$ total nodes. Let $x_1, \ldots, x_m \in \mathbb{K}_1$ be the prescribed sampling device and $u \in \mathbb{V}$. Then, define ψ_u as follows:

$$\psi_u : [0,1]^n \to \mathbb{R}, \quad \mathbf{y} \mapsto \psi\left(u(x_1), \ldots, u(x_m), \mathbf{y}\right).$$

Since $u(x_1), \ldots, u(x_m)$ can be added onto the first layer's bias term, ψ_u is a NN with n inputs, $\leq k$ layers, and $\leq 2^k$ total nodes. The second statement of the theorem holds by Theorem 1.1 of Telgarsky (2016), as the ReLU activation function is a $(1,1,1)$-semi-algebraic gate. □

Theorem 1 illustrates that increasing the depth of a NN can make operator approximation much less expensive. This suggests that UATs for deep operator NNs comprise an important contribution to our understanding of the limitations of deep learning and expressibility of nonlinear operators.

3 Construction of the Deep Narrow Operator Neural Network

We present two results on the existence of a deep NN approximation of a nonlinear continuous operator. One is an explicit reconnection of an existing wide NN and the other is an abstract existence argument. In this section, $\mathbb{X}$ is a Banach space, and $\mathbb{K}_1 \subset \mathbb{X}$ is compact. Let $\mathbb{V} \subset \mathcal{C}(\mathbb{K}_1) := \mathcal{C}(\mathbb{K}_1, \mathbb{R})$ be compact in $\mathcal{C}(\mathbb{K}_1)$, which is equipped with the topology induced by the uniform norm. Suppose that $n \in \mathbb{N}$, $\mathbb{K}_2 \subset \mathbb{R}^n$ is compact, and $G : \mathbb{V} \to \mathcal{C}(\mathbb{K}_2)$ is a nonlinear continuous operator. In Chen & Chen (1995), it is shown that G can be uniformly approximated by a 4-layer NN if the activation function has the density property. More precisely, given any $\varepsilon > 0$, there are positive integers $M, N, m \in \mathbb{N}$, real numbers $c_i^k, \zeta_k, \xi_{ij}^k \in \mathbb{R}$, vectors $\boldsymbol{\omega}_k \in \mathbb{R}^n$, and sensors $x_j \in \mathbb{K}_1$ such that

$$\left| G(u)(\mathbf{y}) - \sum_{k=1}^{N} \left[\sum_{i=1}^{M} c_i^k \sigma\left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right) \right] \sigma(\boldsymbol{\omega}_k \cdot \mathbf{y} + \zeta_k) \right| < \varepsilon,$$

for all $u \in \mathbb{V}$ and $\mathbf{y} \in \mathbb{K}_2$. The architecture of this NN is shown in Figure 1 (left). The input layer consists of $\mathbf{y} = (y_1, \ldots, y_n)$ and $(u_1, \ldots, u_m) = (u(x_1), \ldots, u(x_m))$. The second layer computes $p_i^k = \sigma\Big(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k\Big)$. The third layer computes $r^k = \sigma(\boldsymbol{\omega}_k \cdot \mathbf{y} + \zeta_k)$ and $q^k = \sum_{i=1}^{M} c_i^k p_i^k$. The fourth layer consists of multiplication neurons that compute $s^k = r^k q^k$ for $k = 1, \ldots, N$, whose sum is the output of the NN.

3.1 Register-Compute Neural Networks

In a fully connected feedforward NN, connections between non-consecutive layers are not allowed. Such NNs are "memoryless," as a neuron in the j^{th} layer receives no input other than the output from the $(j-1)^{\text{th}}$ layer. One can introduce memory into a NN by showing that a neuron with a particular activation function, weights, and bias can uniformly approximate the identity function on a compact set Kidger & Lyons (2019). We use such neurons to propagate the inputs through the layers of our NN to use them in later computations. This motivates the following definition of the basic model in our construction.

Definition 1. *Let $p, q \in \mathbb{N}$. A (p,q)-register-compute NN is a fully connected feedforward NN with $p+q$ neurons in each hidden layer. In each layer, p neurons are called registers, ordered so that the only nonzero weight in the j^{th} register of layer i is from the output of the j^{th} register of layer $i-1$.*

Although all pairs of neurons in consecutive layers are connected in a fully connected feedforward NN, we effectively "disconnect" non-corresponding registers by setting the weights to be zero.

3.2 Constructing the First Deep Operator NN: Reconnecting the Wide Operator NN

We observe that neurons in each hidden layer of the operator NN in Chen & Chen (1995) can be moved one-by-one into different hidden layers. Moreover, if σ has the properties in Theorem 2, then a σ-activated neuron can be used to uniformly approximate the identity map $\iota_{\mathbb{K}}$ on any compact set $\mathbb{K} \subset \mathbb{R}$ (Lemma 4.1 of Kidger & Lyons (2019)). This allows us to propagate inputs from one layer

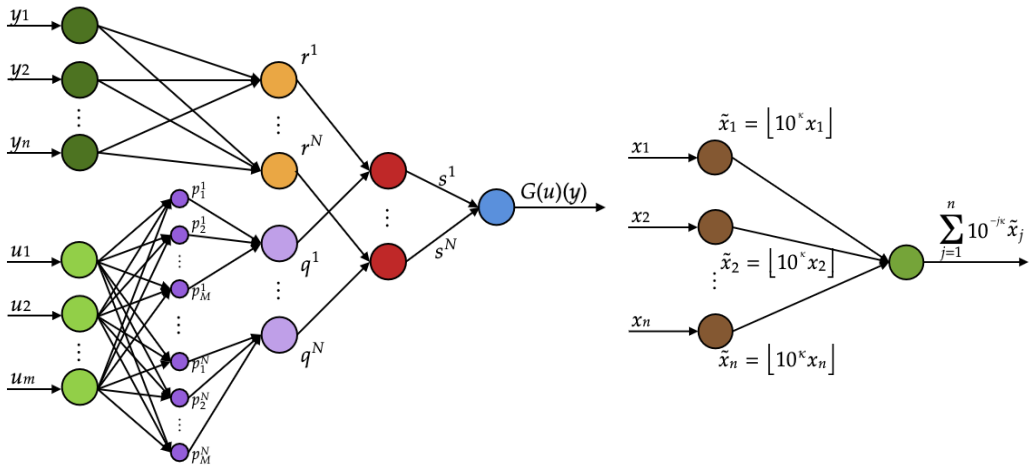

Figure 1: Left: The wide operator NN from Chen & Chen (1995). Right: Our encoder model.

to the next. By rearranging the neurons in the shallow, wide operator NN, we get a deep NN whose width depends only on the size of the input layer.

Theorem 2. *Let $\sigma : \mathbb{R} \to \mathbb{R}$ have the density property. Suppose that σ is also continuously differentiable at one or more points with a nonzero derivative. Then, for any $\varepsilon > 0$, there exists a function $F : \mathbb{R}^{m+n} \to \mathbb{R}$ represented by a σ-activated NN of width at most $m + n + 5$ such that*

$$|G(u)(\mathbf{y}) - F(u(x_1), \ldots, u(x_m), \mathbf{y})| < \varepsilon$$

for all $u \in \mathbb{V}$ and $\mathbf{y} \in \mathbb{K}_2$. Moreover, if a σ-activated NN of width 3 and depth L approximates the multiplication map $(a, b) \mapsto ab$ on any compact set up to any uniform error, then the network F has depth in $\mathcal{O}((M + L)N)$, where $M, N, m, \{x_j\}_{j=1}^m$ are as in Theorem 5 of Chen & Chen (1995).

Proof. Let $H : \mathbb{R}^{m+n} \to \mathbb{R}$ be the function given by the NN in Theorem 5 of Chen & Chen (1995) that approximates the operator G to within $\varepsilon/5$. We construct an $(m + n, 5)$-register-compute NN F with $m + n$ inputs, one output, and $(M + L + 1)N + 1$ layers, where L is a positive integer defined later in equation a. Among the 5 neurons that are not registers in each hidden layer, 1 neuron is referred to as the *output augmenter*. 2 neurons are referred to as the *adder* 1 and the *adder* 2, respectively, and the remaining 2 neurons are referred to as the *computation neurons*.

The $m + n$ input layer values are passed into the corresponding $m + n$ registers in the first hidden layer. A register that receives a value $u(x_j)$ is called a u-register. If this register is in the kth hidden layer, then we denote its output by u_j^k. A y-register and its output y_j^k are similarly defined. We also define $u_j^0 = u(x_j)$ and $y_j^0 = y_j$. Up to a small error so that ε_4 in equation 2 satisfies $|\varepsilon_4| < \varepsilon/5$ for all $u \in \mathbb{V}, \mathbf{y} \in \mathbb{K}_2$, each register computes a function that is close to the identity function $\iota_{\mathbb{L}}$ in $L^\infty(\mathbb{L})$, where $\mathbb{L}$ is the range of the output of the previous register.

We further divide the $(M + L + 1)N$ hidden layers into N sections of $M + L + 1$ layers. In the kth section, the ith adder 1 in each layer computes:

$$p_i^k = \sigma\left(\sum_{j=1}^{m} \xi_{ij}^k u_j^{(M+L+1)(k-1)+i-1} + \theta_j^k\right), \qquad 1 \le i \le M$$

using the outputs of the u-registers $u_j^{(M+L+1)(k-1)+i-1}$ from the previous hidden layer.

Let $\tilde{q}_i^k = c_i^k p_{i-1}^k + q_{i-1}^k$, where q_{i-1}^k is the output of the $(i-1)$th adder 2 in the kth section. We set $p_0^k = q_0^k = 0$. The affine transformation of the ith adder 2 in kth section computes $\tilde{q}_i^k$ and, together with the activation function, propagates $\tilde{q}_i^k$ using the identity approximation mentioned above, up to a small error so that ε_3 in equation 1 satisfies $|\varepsilon_3| < \varepsilon/5$ for all $u \in \mathbb{V}, \mathbf{y} \in \mathbb{K}_2$. The output is then denoted by q_i^k.

The $(M+1)$th adder 1 in the kth section computes:

$$r^k = \sigma(\boldsymbol{\omega}_k \cdot \mathbf{y}^{(M+L+1)(k-1)+M} + \zeta_k)$$

using the outputs of the y-registers of the previous hidden layer:

$$\mathbf{y}^{(M+L+1)(k-1)+M} = (y_1^{(M+L+1)(k-1)+M}, \ldots, y_n^{(M+L+1)(k-1)+M}),$$

The $(M+1)$th adder 2 in the kth section propagates q_M^k, and its output is denoted by q^k.

In the next L_k layers, we can use adder 1, adder 2, and the 2 computation neurons to approximate $r^k q^k$ up to an error so that ε_2 in equation 1 satisfies $|\varepsilon_2| < \varepsilon/5$ for all $u \in \mathbb{V}, \mathbf{y} \in \mathbb{K}_2$ (Proposition 4.9 of Kidger & Lyons (2019)). This number, denoted by s^k, is then added to the output augmenter which, unless otherwise stated, propagates the value from the previous layer, up to an error so that ε_1 in equation 1 satisfies $|\varepsilon_1| < \varepsilon/5$ for all $u \in \mathbb{V}, \mathbf{y} \in \mathbb{K}_2$. The initial value in the output augmenter is set to 0.

We set

$$L := \max_{1 \le k \le N} L_k. \tag{a}$$

We assume that any neurons from layer $L_k + 1$ to L in the kth section do nothing but propagate the values from the previous hidden layer. Once the Nth section is computed, we add s^N to the augmenter. Now, the value of the augmenter in the $((M+L)N+1)$th layer is given by $S_{u,\mathbf{y}}$, where

$$S_{u,\mathbf{y}} = \sum_{k=1}^{N} s^k + \varepsilon_1 = \sum_{k=1}^{N} q^k r^k + \varepsilon_1 + \varepsilon_2 = \sum_{k=1}^{N} \left(\sum_{i=1}^{M} c_i^k p_{i-1}^k \right) r^k + \varepsilon_1 + \varepsilon_2 + \varepsilon_3 \tag{1}$$

$$= \sum_{k=1}^{N} \left[\sum_{i=1}^{M} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{ij}^k u_j^{\ell(k)+i-1} + \theta_j^k \right) \right] g(\boldsymbol{\omega}_k \cdot \mathbf{y}^{\ell(k)+M} + \zeta_k) + \varepsilon_1 + \varepsilon_2 + \varepsilon_3$$

$$= \sum_{k=1}^{N} \left[\sum_{i=1}^{M} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_j^k \right) \right] g(\boldsymbol{\omega}_k \cdot \mathbf{y} + \zeta_k) + \varepsilon_1 + \varepsilon_2 + \varepsilon_3 + \varepsilon_4 \tag{2}$$

$$= H(u(x_1), \ldots, u(x_m), \mathbf{y}) + \varepsilon_1 + \varepsilon_2 + \varepsilon_3 + \varepsilon_4, \tag{3}$$

where $\ell(k) = (M+L+1)(k-1)$. Since $|\varepsilon_j| < \varepsilon/5$ for $j = 1, 2, 3, 4$, we have

$$|G(u)(\mathbf{y}) - S_{u,\mathbf{y}}| \le |G(u)(\mathbf{y}) - H(u(x_1), \ldots, u(x_j), \mathbf{y})| + |(H(u(x_1), \ldots, u(x_j), \mathbf{y}) - S_{u,\mathbf{y}}|$$
$$\le \frac{\varepsilon}{5} + |\varepsilon_1| + |\varepsilon_2| + |\varepsilon_3| + |\varepsilon_4| < \varepsilon$$

for all $u \in \mathbb{V}, \mathbf{y} \in \mathbb{K}_2$. The result follows as $S_{u,\mathbf{y}} = F(u(x_1), \ldots, u(x_m), \mathbf{y})$. □

Theorem 2 is a theoretical guarantee that a deep NN can approximate the operator G. In particular, the width of our NN does not depend on M and N in Theorem 5 in Chen & Chen (1995), where these parameters are obtained abstractly and do not have intuitive interpretations.

Theorem 2 has two shortcomings. First, the total number of neurons in the deep operator NN in Theorem 2 is $\Omega((m+n)(M+L)N)$, whereas that of a shallow, wide NN in Chen & Chen (1995) is $\mathcal{O}(m+n+MN)$. We emphasize, however, that the NN in Theorem 2 is not necessarily the simplest one to achieve an ε-approximation. In fact, deep NNs can outperform the shallow ones in approximating certain operators (see Section 2).

Second, the NN's width depends on m, which in turn depends on ε. Thus, while we have eliminated the dependence of the width on M and N, the number of sensors is reflected in the width, and a large sampling device is needed to achieve an accurate approximation, making the NN both arbitrarily deep and arbitrarily wide. To address this, we have two avenues. First, we find a relationship between m and ε. Proving some rate of growth of m with respect to ε would make Theorem 2 more informative, as in Lu et al. (2019), for example. However, results of this type are in a more specific context, and a relationship between m and ε in the general setting is challenging. Also, to prevent the width of our deep NN from growing too fast as $\varepsilon \to 0$, we would like to have $m(\varepsilon) = \mathcal{O}(\log(1/\varepsilon))$. So far, we are not aware of any existing result that demonstrates that $m(\varepsilon) = \mathcal{O}(\log(1/\varepsilon))$ is possible. Instead, we need to find a way to reduce the width of the network, regardless of the number of inputs.

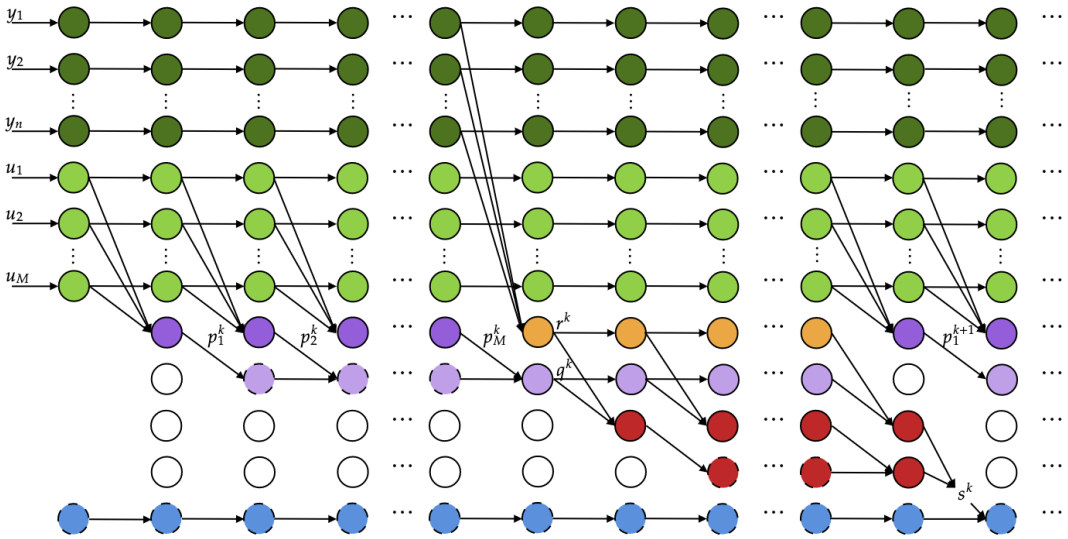

Figure 2: A portion of the deep NN developed by transforming the wide operator NN from Chen & Chen (1995) into a register model. The process can be found in Kidger & Lyons (2019).

3.3 Input Encoding and Reduction

It has been shown that arbitrarily-deep NNs of width $m + 3$ can uniformly approximate functions $f : \mathbb{K} \to \mathbb{R}$, where $\mathbb{K} \subset \mathbb{R}^m$ is a compact set (Kidger & Lyons, 2019). We further reduce this width to a constant, eliminating the dependence on m. However, it is known that for certain activation functions, the width m is not enough to uniformly approximate continuous functions on compact sets (Hanin & Sellke, 2017; Lu et al., 2017). Therefore, we slightly modify the architecture of the NNs to make them more flexible.

In Kidger & Lyons (2019), m inputs are propagated throughout the entire network, which requires m neurons in each hidden layer. Our trick is to truncate the inputs using the floor function and then encode them into a single neuron. This single neuron is then propagated using only one neuron from one hidden layer to the next and is decoded when necessary. When decoding, the inputs are decoded one-by-one, and then we immediately pass the decoded value into the computation neurons.

We first define terminology for truncating and encoding inputs. A *truncation neuron* takes an input x and produces the output $\lfloor 10^\kappa x \rfloor$, where κ is an arbitrary integer. A *NN with truncated inputs* is a NN where every input has been passed through a truncation neuron. The *width* of the NN with truncated inputs is the size of the largest hidden layer, ignoring the truncation neurons applied immediately to the input layer. Therefore, a (p, q)*-register-compute* NN with truncated inputs is a NN with truncated inputs that is a (p, q)-register-compute NN if the outputs of the truncation layer are viewed as the inputs of the NN. A σ*-activated* NN with truncated inputs is a NN whose neurons in all hidden layers have σ as the activation function, ignoring the truncation neurons applied immediately to the input layer.

Theorem 3. *Suppose σ is non-polynomial and continuously differentiable at one or more points with nonzero derivative. Let $\mathbb{K} \subset \mathbb{R}^n$ be a compact set, and let $f : \mathbb{K} \to \mathbb{R}$ be continuous. For each $\varepsilon > 0$, there is a function $g : \mathbb{R}^n \to \mathbb{R}$ represented by a σ-activated NN with truncated inputs of width 5 such that*

$$|f(\mathbf{x}) - g(\mathbf{x})| < \varepsilon, \qquad \mathbf{x} \in \mathbb{K}.$$

Proof. Without loss of generality, we let $\mathbb{K} \subset (0, 1)^n$. Otherwise, we scale and translate the domain with the truncation neuron and bias terms in the first hidden layer. There is an identity-activated $(n, 2)$-register-compute NN, $h : \mathbb{R}^n \to \mathbb{R}$, such that $|h(\mathbf{x}) - f(\mathbf{x})| < \varepsilon/3$ for all $\mathbf{x} \in \mathbb{K}$ (Kidger & Lyons, 2019). Each of the n inputs is passed into a unique register in the first hidden layer, then propagated by the corresponding register in each hidden layer. Among the two remaining non-register neurons in each hidden layer, one neuron is the *computation neuron*, which applies an

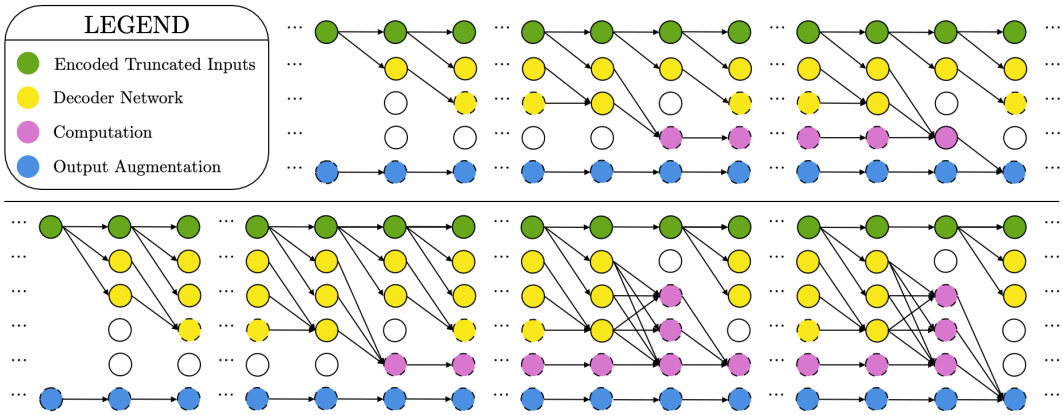

Figure 3: Our decoder model for non-polynomial σ (top) and polynomial σ (bottom).

affine transformation to the outputs of the registers in the previous hidden layer. The other neuron is the *augmentation neuron*, which sums the outputs of the computation neuron and the augmentation neuron in the previous layer. The output of the first augmentation neuron is set to zero.

Since the NN h is identity-activated, it can be restructured so that each computation neuron only reads one input from the registers and its own output from the previous layer, and applies an affine transformation. To see this, let $\mathbf{x} \mapsto \sum_{j=1}^{\ell} w_j x_j + b$ be a computation neuron. We replace the layer of this neuron by ℓ layers and use the computation neuron from each of the ℓ layers to compute $w_1 x_1, w_1 x_1 + w_2 x_2, \ldots, \sum_{j=1}^{\ell-1} w_j x_j, \sum_{j=1}^{\ell} w_j x_j + b$, respectively. Each of the remaining neurons in the ℓ layers applies the identity to the corresponding output from the previous layer. Let $L+1$ be the depth of this restructured NN.

We now show how we can store input approximations in a single neuron. For large $\kappa \in \mathbb{N}$, we let $\tilde{x}_j = \lfloor 10^{\kappa} x_j \rfloor$ for $1 \leq j \leq n$ be the truncated inputs. The register in the first hidden layer computes

$$r := \sum_{j=1}^{n} 10^{-j\kappa} \tilde{x}_j = 10^{-\kappa} \tilde{x}_1 + 10^{-2\kappa} \tilde{x}_2 + \cdots + 10^{-n\kappa} \tilde{x}_n,$$

where the remaining registers take the input r and pass it as the output. Now, we define a series of *decoder functions*, $\varphi_1, \ldots, \varphi_n$. Every $a = 10^{-n\kappa} M \in [0, 1), M \in \mathbb{N}_0$ can be expanded uniquely as

$$a = a_1 10^{-\kappa} + a_2 10^{-2\kappa} + \cdots + a_m 10^{-n\kappa},$$

where $a_1, \ldots, a_m$ are integers in $[0, 10^{\kappa}]$. We set $\varphi_j([a - 10^{-n\kappa-1}, a + 10^{-n\kappa-1}]) := 10^{-n\kappa} a_j$ for each a and then extend φ_j to the interval $[0, 1]$ continuously by the Tietze Extension Theorem.

Now, we construct a $(1, 4)$-register-compute NN with truncated inputs. We let $p : \mathbb{R}^n \to \mathbb{R}$ be the function it represents. Unlike most fully connected feedforward NNs, the neurons in each layer have different activation functions. The register uses the identity activation function. Among the four remaining neurons in each hidden layer, one neuron is called the *computation neuron*, and one neuron is called the *augmentation neuron*, which uses the identity activation function. The remaining two neurons are called the *decoder neurons*, which use σ as the activation function.

Let $i \in \{1, \ldots, L\}$. In the NN h, by assumption, only one of $x_1, \ldots, x_n$, say x_j, is read by the computation neuron in the ith layer. We construct the NN p by building $L + 1$ chunks, where the last chunk is the output layer. To construct the ith chunk, we use the two decoder neurons from each hidden layer together with the register to approximate $\varphi_j(r)$ up to a small error, as in Proposition 4.9 of Kidger & Lyons (2019). We note that $\varphi_j(r) = 10^{-\kappa} \tilde{x}_j \approx x_j$. This decoded value is then passed into the computation neuron for the affine transformation done at the ith layer in the NN h.

Compared to h, the difference in the output of p is induced by two steps: the truncating x to obtain $10^{-\kappa} \tilde{x}$, and decoding to obtain an approximation of $\phi_j(r) = 10^{-\kappa} \tilde{x}$. The first error can be made arbitrarily small by taking κ large enough and the second error can also be made arbitrarily small as in the previous paragraph. Thus, we can construct the NN p so that $|p(\mathbf{x}) - h(\mathbf{x})| < \varepsilon/3$ for $\mathbf{x} \in \mathbb{K}$.

It remains to construct a NN with truncated inputs that only uses σ as the activation function. To do so, we define a NN $g : \mathbb{R}^n \to \mathbb{R}$ whose architecture completely inherits that of p, except the registers, computation neurons, and augmentation neurons are σ-activated. As before, we may use a σ-activated neuron to mimic the identity activation function. For the register, we make the approximate identity accurate enough so that the perturbed value of a, denoted by $\tilde{a}$, always satisfies $|a - \tilde{a}| < 10^{-n\kappa-1}$. Hence, we have that $\varphi_j(\tilde{a}) \equiv \varphi_j(a)$ throughout the entire NN. Since the values in the computation neurons and the augmentation neurons can be propagated arbitrarily accurately, we have $|g(\mathbf{x}) - p(\mathbf{x})| < \varepsilon/3$ for $\mathbf{x} \in \mathbb{K}$. Therefore, we have

$$|g(\mathbf{x}) - f(\mathbf{x})| \leq |g(\mathbf{x}) - p(\mathbf{x})| + |p(\mathbf{x}) - h(\mathbf{x})| + |h(\mathbf{x}) - f(\mathbf{x})| < \varepsilon, \qquad \mathbf{x} \in \mathbb{K}.$$

□

Theorem 3 shows that truncating inputs allows any continuous function on a compact set to be uniformly approximated by deep NNs of constant width. This independence of width and dimension overcomes the problematic growth of the size of the sampling device in Lu et al. (2019). Since non-affine polynomial activation functions satisfy the arbitrary-depth UAT, we obtain Theorem 4. Analogous to Theorem 3, which extends Proposition 4.9 of Kidger & Lyons (2019), Theorem 4 naturally extends Proposition 4.11 of Kidger & Lyons (2019). As opposed to wide NNs, deep NNs with (non-affine) polynomial activation functions approximate continuous functions nicely.

Theorem 4. *Let $\mathbb{K} \subset \mathbb{R}^n$ be a compact set and $f : \mathbb{K} \to \mathbb{R}$ be a continuous function. For each $\varepsilon > 0$, there is a function $g : \mathbb{R}^n \to \mathbb{R}$ represented by a σ-activated NN with truncated inputs of width 6 such that*

$$|f(\mathbf{x}) - g(\mathbf{x})| < \varepsilon, \qquad \mathbf{x} \in \mathbb{K}.$$

Proof. Without loss of generality, let $\mathbb{K} \subset (0, 1)^n$. Let $p = \sum_{j=1}^{\ell} p_j : \mathbb{K} \to \mathbb{R}$ be a polynomial with monomials p_j such that $|f(\mathbf{x}) - p(\mathbf{x})| < \varepsilon/3$ for $\mathbf{x} \in \mathbb{K}$. There is a $(n, 4)$-register-compute NN $h : \mathbb{R}^n \to \mathbb{R}$ satisfying $|h(\mathbf{x}) - p(\mathbf{x})| < \varepsilon/3$ for $\mathbf{x} \in \mathbb{K}$, where each hidden layer contains n identity-activated registers that propagate the n inputs, one σ-activated augmentation neuron that stores the output and never takes any register as an input, and three σ-activated computation neurons that compute the monomials p_j (Proposition 4.6 and Proposition 4.11 in Kidger & Lyons (2019)).

The computation neurons take no more than one value from the registers as the input in each layer. Moreover, when these neurons need inputs from one of the registers, the outputs of all but possibly one in the previous layer do not become the input of any other neurons in the current layer.

As in the proof of the Theorem 3, we can construct a σ-activated NN $g : \mathbb{R}^n \to \mathbb{R}$ with truncated input of width 8 such that $|h(\mathbf{x}) - g(\mathbf{x})| < \varepsilon/3$, for $\mathbf{x} \in \mathbb{K}$. In particular, each hidden layer contains one register that propagates the encoded input as in the proof of Theorem 3, three decoder neurons that, together with the register, approximate the decoders $\varphi_1, \ldots, \varphi_n$ (Proposition 4.11 in Kidger & Lyons (2019)), 3 computations neurons as in h, and 1 augmentation neuron as in h.

Finally, when a decoder neuron is activated and its output becomes an input to the next layer, the computation neurons read the input from the register. However, when the computation neurons read inputs from the register, the outputs of two of them in the previous layer are not used in the current layer. Thus, two computation neurons can be reused in the architecture of the decoder. Hence, only $3+3-2 = 4$ neurons are used to implement the decoder and the computation unit, and consequently, g can be realized by a NN with truncated inputs of width 6. Now, for all $\mathbf{x} \in \mathbb{K}$, we have

$$|g(\mathbf{x}) - f(\mathbf{x})| \leq |g(\mathbf{x}) - h(\mathbf{x})| + |h(\mathbf{x}) - p(\mathbf{x})| + |p(\mathbf{x}) - f(\mathbf{x})| < \varepsilon.$$

□

We note that the success of the encoder/decoder does not depend on the representation being decimal. They can be equivalently constructed using binary representations of numbers, so that if the operation in the truncation neuron is $x \mapsto \lfloor 2^\kappa x \rfloor$, Theorem 3 and Theorem 4 still hold. This result is more relevant to most modern machines, which store floating-point numbers with finitely many bits, conduct floating-point arithmetic in binary, and perform $x \mapsto 2^\kappa x$ easily.

3.4 CONSTRUCTING THE SECOND DEEP OPERATOR NN: AN ABSTRACT APPROACH

Now, we have the tools to eliminate the dependence of the NN's width on the size of the sampling device. We adopt an abstract strategy to construct an operator NN with truncation whose width is a constant. To do so, we view the NN in Chen & Chen (1995) as a function f from $\mathbb{R}^{m+n}$ to $\mathbb{R}$, for we encode the input function u as m values. Therefore, to approximate the operator G, it suffices to approximate f uniformly.

Theorem 5. *Let σ be non-polynomial (resp. non-affine), continuously differentiable at one or more points with nonzero derivative. Then, for every $\varepsilon > 0$, there are points $x_1, \dots, x_m \in \mathbb{K}_1$ and a function $F : \mathbb{R}^{m+n} \to \mathbb{R}$ given by a σ-activated NN with truncated inputs of width 5 (resp. 6), such that*

$$|G(u)(\mathbf{y}) - F(u(x_1), \dots, u(x_m), \mathbf{y})| < \varepsilon$$

for all $u \in \mathbb{V}$ and $\mathbf{y} \in \mathbb{K}_2$. Moreover, m is independent of σ.

Proof. Define $\mathbb{U}_j = \{u(x_j) \mid u \in \mathbb{V}\}$ and $\mathbb{U} = \prod_{j=1}^m \mathbb{U}_j$. The evaluation map $\phi_j : \mathbb{V} \to \mathbb{R}, u \mapsto u(x_j)$ is continuous. Hence, $\mathbb{U}_j = \phi_j(\mathbb{V})$ is compact for each j, and so is $\mathbb{U}$ by Tychonoff's Theorem. Let $g : \mathbb{R} \to \mathbb{R}$ be an arbitrary function with the density property. This function induces points $x_1, \dots, x_m \in \mathbb{K}_1$ and a function $H : \mathbb{U} \times \mathbb{K}_2 \to \mathbb{R}$ such that

$$|G(u)(\mathbf{y}) - H(u(x_1), \dots, u(x_m), \mathbf{y})| < \varepsilon/2$$

for any $u \in \mathbb{V}$ and $\mathbf{y} \in \mathbb{K}_2$ (Theorem 5 in Chen & Chen (1995)). Let $F : \mathbb{R}^{m+n} \to \mathbb{R}$ be the function represented by the NN with truncated inputs constructed in Theorem 3 or Theorem 4 associated with the function H and the approximation error $\varepsilon/2$. The statement of the theorem follows from the triangle inequality and the fact that g is arbitrary, making m independent of δ. □

Compared to Theorem 2, Theorem 5 gives us a deep operator NN whose width is constant. Moreover, it allows us to use non-affine polynomial activation functions, which are known to be powerless in approximating using the 2-layer networks (Pinkus, 1999). Inspired by Proposition 4.17 of Kidger & Lyons (2019), we have the following extension of Theorem 5.

Corollary 6. *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a polynomial such that $\sigma'(\alpha) = 0$ and $\sigma''(\alpha) \neq 0$ for some $\alpha \in \mathbb{R}$. Then, for every $\varepsilon > 0$, there exist points $x_1, \dots, x_m \in \mathbb{K}_1$ and a function $F : \mathbb{R}^{m+n} \to \mathbb{R}$ represented by a σ-activated NN with truncated inputs of width 5, such that*

$$|G(u)(\mathbf{y}) - F(u(x_1), \dots, u(x_m), \mathbf{y})| < \varepsilon, \qquad u \in \mathbb{V}, \quad \mathbf{y} \in \mathbb{K}_2.$$

Proof. The proof follows from Theorem 4. The NN h in Theorem 4 can be implemented using a $(m, 3)$-register-compute NN. Two neurons can implement the decoder in g. When the decoder is activated, it uses one of the two computation neurons. The rest of the proof is then analogous to the proof of Theorem 5 and the width of g is $2 + 2 + 2 - 1 = 5$, where the first "2" corresponds to the register and the output augmenter. The second and the third "2"s are the number of neurons needed to implement the decoder and the number of computation neurons, respectively. □

Corollary 6 in combination with the non-polynomial σ case means that "most" activation functions require our NN with truncated inputs to have a width of 5. This is a slight improvement compared to Theorem 5, in which we require a width of 6 when σ is a non-affine polynomial.

4 CONCLUSION

This paper proves that arbitrary-depth operator NNs with a large class of activation functions are universal approximators. Our main theorem is a UAT for operator NNs of width 5 with a non-polynomial that is continuously differentiable at a point with nonzero derivative (see Theorem 5). Our proof technique is robust enough to handle non-affine polynomial activation functions too (see Theorem 4 and Corollary 6). We also construct an operator ReLU NN of depth $2k^3 + 8$ and constant width that cannot be well-approximated by any operator ReLU NN of depth k, unless its width is exponential in k (see Theorem 1). This demonstrates that deep, narrow NNs are better than shallow, wide ones at approximating certain continuous nonlinear operators. We hope that this adds theoretical justification to those that use deep operator NNs.

REFERENCES

Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017.

Shengze Cai, Zhicheng Wang, Lu Lu, Tamer A Zaki, and George Em Karniadakis. DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *Journal of Computational Physics*, 436:110296, 2021.

Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995. doi: 10.1109/72.392253.

Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on learning theory*, pp. 907–940. PMLR, 2016.

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

Patrick Kidger and Terry J. Lyons. Universal approximation with deep narrow networks. *CoRR*, abs/1905.08539, 2019. URL http://arxiv.org/abs/1905.08539.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

Samuel Lanthaler, Siddhartha Mishra, and George Em Karniadakis. Error estimates for DeepOnets: A deep learning framework in infinite dimensions. *arXiv preprint arXiv:2102.09618*, 2021.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6232–6240, 2017.

Allan Pinkus. Approximation Theory of the MLP Model in Neural Networks. *Acta Numerica*, 8: 143–195, 1999. doi: 10.1017/S0962492900002919.

Fabrice Rossi, Nicolas Delannay, Brieuc Conan-Guez, and Michel Verleysen. Representation of functional data in neural networks. *Neurocomputing*, 64:183–210, 2005.

Zuowei Shen, Haizhao Yang, and Shijun Zhang. Neural network approximation: Three hidden layers are enough. *Neural Networks*, 141:160–173, Sep 2021a. ISSN 0893-6080. doi: 10.1016/j.neunet.2021.04.011. URL http://dx.doi.org/10.1016/j.neunet.2021.04.011.

Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation: Achieving arbitrary accuracy with a fixed number of neurons, 2021b.

Matus Telgarsky. Benefits of depth in neural networks. *CoRR*, abs/1602.04485, 2016. URL http://arxiv.org/abs/1602.04485.