A Deep Learning Framework for Multi-Operator Learning: Architectures and Approximation Theory

Adrien Weihs[1], Jingmin Sun[2], Zecheng Zhang[3], and Hayden Schaeffer[1]

[1]Department of Mathematics,
University of California Los Angeles,
Los Angeles, CA 90095, USA.

[2]Department of Applied Mathematics and Statistics,
Johns Hopkins University,
Baltimore, MD 21218 , USA.

[3]Department of Applied Computational Mathematics and Statistics,
University of Notre Dame,
Notre Dame, IN 46556, USA

October 2025

Abstract

While many problems in machine learning focus on learning mappings between finite-dimensional spaces, scientific applications require approximating mappings between function spaces, i.e., operators. We study the problem of learning collections of operators and provide both theoretical and empirical advances. We distinguish between two regimes: (i) multiple operator learning, where a single network represents a continuum of operators parameterized by a parametric function, and (ii) learning several distinct single operators, where each operator is learned independently. For the multiple operator case, we introduce two new architectures, MNO and MONet, and establish universal approximation results in three settings: continuous, integrable, or Lipschitz operators. For the latter, we further derive explicit scaling laws that quantify how the network size must grow to achieve a target approximation accuracy. For learning several single operators, we develop a framework for balancing architectural complexity across subnetworks and show how approximation order determines computational efficiency. Empirical experiments on parametric PDE benchmarks confirm the strong expressive power and efficiency of the proposed architectures. Overall, this work establishes a unified theoretical and practical foundation for scalable neural operator learning across multiple operators.

Keywords and phrases. Deep Neural Networks, Approximation Theory, Neural Scaling Laws, Operator Learning, Multi-Operator Learning.
Mathematics Subject Classification. 41A99, 68T07

1 Introduction

Classical machine learning is primarily concerned with learning functions of the form

$$f : \mathbb{R}^n \to \mathbb{R}^d$$

where finite-dimensional inputs are mapped to finite-dimensional outputs. In many scientific and engineering applications, however, the goal is to approximate mappings between function spaces,

$$G : U \to V$$

where U and V are typically subsets of infinite-dimensional Banach or Hilbert spaces. Such problems arise, for instance, in learning solution operators of ordinary and partial differential equations [4,12,39,45], and span a wide range of scientific and engineering domains [10,24,37,51,53,65]. This framework is known as operator learning, and it extends classical supervised learning from the setting of functions to that of operators acting on functions. We refer to [32,46] and references therein for a review and comparison of approaches in this topic.

Neural networks form a natural framework for operator learning, combining the flexibility to approximate complex nonlinear mappings with a strong record of empirical success in scientific and engineering applications [18–20, 29, 64]. Modern operator-learning networks [11, 12] typically decompose the learned operator into interacting subnetworks that process different aspects of the input, such as spatial variables, input functions, or parameters, before combining them through summation or tensor-like contractions. For example, in the terminology of DeepONet [45], the branch subnetwork encodes the input function, while the trunk subnetwork represents a basis for the output function space. This basis, constructed by neural networks, can also be designed to mimic classical finite element bases. These architectures draw inspiration from low-rank approximations, where complex mappings are expressed as sums of separable, lower-dimensional functions [48]. Neural operator networks can thus be viewed as nonlinear analogues of such expansions, with each subnetwork learning one component of a functional basis.

However, the neural network approach also introduces challenges that are inherent to its design: in particular, how to construct architectures that are both simple to implement and empirically effective, while also supported by rigorous mathematical guarantees. These difficulties become especially pronounced in the multi-operator setting, where the need to represent numerous complex operators often leads to rapidly increasing architectural complexity.

In this work, we distinguish between two related but conceptually distinct settings in which numerous operators are involved. The first is the multiple operator learning setting, $G : W \to \{G[\alpha] : U^{(\alpha)} \mapsto V^{(\alpha)}\}_{\alpha \in W}$, where the parametric function $\alpha \in W$ serves as an explicit input to the network, allowing a single model to represent a continuum of operators indexed by α. The second concerns learning several single operators, $\{G^{(j)} : U^{(j)} \to V^{(j)}\}_{j \in J}$, where each operator is learned independently and the dependence on the index j remains external to the model. We summarize and contrast the main differences between these two formulations in Table 4. This distinction clarifies the different modeling challenges posed by operator learning in practice. Building on this, we investigate fundamental theoretical and practical aspects of designing expressive and efficient neural architectures for learning collections of operators. Specifically, we address three central questions:

Q.1 Can one construct architectures that are provably expressive, yielding (quantitative) universal approximation guarantees?

Q.2 How can network architectures be designed to exploit shared structure across related operators, balance complexity among functional components, and attain optimal approximation and scaling performance?

Q.3 Are the proposed architectures empirically efficient and capable of strong performance on representative learning tasks?

In addressing these questions, we provide both theoretical insights and empirical evidence that clarify the principles underlying expressive and scalable operator-learning networks.

1.1 Key Contributions

Our main contributions in the multiple operator learning setting (summarized in Table 1) are as follows:

1. We introduce two new architectures for multiple operator learning, MNO and MONet, designed to generalize existing operator learning models and provide flexible building blocks for theoretical and practical analysis.

2. We establish **universal approximation** results for multi-operator learning, showing that both architectures can approximate any continuous operator to arbitrary accuracy on compact sets.

3. We establish a **weak universal approximation** property for multi-operator learning, proving that both architectures can approximate measurable operators, thereby extending expressivity guarantees beyond the continuous setting.

4. We establish a **strong universal approximation** property for Lipschitz operators when approximated using our proposed MNO model for multi-operator learning. In this case we also derive **scaling laws**, i.e., quantitative estimates of the required network size to achieve a prescribed accuracy. Specifically, we show that the approximation error ε scales as follows for $N_\#$ the total number of parameters in the network:

$$\varepsilon \asymp \left(\frac{\log \log N_\#}{\log \log \log N_\#} \right)^{-1/d_W},$$

where d_W denotes the dimension of the domain of functions in W. This is done in the general multi-operator setting without additional knowledge of the properties of the collection operators besides their regularity.

5. We show that our scaling results apply not only to the proposed MNO, but also **extend to a broad family of architectures**, including MIONet [25], thereby unifying several existing approaches under a common framework.

6. We complement our theoretical contributions with **empirical validation** on a wide range of PDE problems, considering both discrete and continuum input parameters α, and demonstrate that the proposed architectures achieve strong performance in practice.

	MONet	MNO
New architectures	✓	✓
Standard UAP (continuous operators)	✓(Theorem 3.4)	✓(Theorem 3.4)
Weak UAP (measurable operators)	✓(Theorem 3.5)	✓(Theorem 3.5)
Strong UAP (Lipschitz operators)	—	✓(Theorem 3.16)
Scaling laws (quantitative rates)	—	✓(Theorem 3.16)
Empirical validation	✓	✓

Table 1: Summary of contributions in the multiple operator learning setting: expressivity guarantees/universal approximation property (UAP), scaling laws, and empirical validation for MNO and MONet which are two architectures for multiple operator learning.

1.2 Additional Contributions

In addition to the main contributions above, our results contribute to the setting of learning several single operators (summarized in Table 2) and are summarized as follows:

1. We establish a **principled framework for selecting architectures** when approximating several single operators, showing that the index dependence j can, under suitable structural conditions on $U^{(j)}$ and $V^{(j)}$, be absorbed into a single network component for improved efficiency. This generalizes to other related works D2NO [63], MIONet [25], and MODNO [61].

2. We demonstrate that the **theoretical approximation order determines how scaling complexity is distributed between subnetworks**, and that the computational burden can be shifted between components without affecting the expressive capacity of the model.

3. We show that the **theoretical approximation order directly impacts efficiency**, emphasizing the key role of architecture in determining computational complexity. Specifically, the attainable rate of approximation with respect to the total number of parameters $N_\#$ depends on the adopted approximation order, yielding either

$$\left(\frac{\log N_\#}{\log\log N_\#}\right)^{-\frac{1}{(1+d_V)d_U}} \quad \text{or} \quad \left(\frac{\log N_\#}{\log\log N_\#}\right)^{-\frac{1}{d_U}},$$

where d_U and d_V denote the dimensions of the domain of functions in U and V, respectively.

Principled framework for architecture design	✓(Remarks 3.9, 3.11, 3.19)
Balancing of scaling complexity across subnetworks	✓(Theorem 3.8 and Remark 3.13)
Impact of approximation order on efficiency	✓(Remarks 3.14 and 3.15)

Table 2: Summary of contributions for the setting of learning several single operators.

Note that the approximation order refers to the choice of hierarchical approximation steps, e.g., approximating functions, then functionals (then operators—in the multiple operator case).

1.3 Related works

A wide range of research has contributed to the development of operator-learning theory and practice, spanning multi-operator learning strategies, neural operator architectures, and the theoretical foundations of expressivity and scaling. We briefly summarize these directions below.

Multi-Operator Learning The motivation for learning collections of operators arises in several contexts: in some applications, it is inherent to the problem formulation itself, while in others, it serves as a means to improve the generalization capability of operator-learning models. Recently, several multi-operator learning approaches have been introduced [7,43,44,49,54,57–61,63]. In particular, the works of [44,54] demonstrated that multi-operator learning can accurately address new tasks beyond those seen during training.

As previously discussed, one can either (1) learn several single operators independently, or (2) consider a more general setting in which the family of operators is encoded through a (discrete or continuous) parametric function α. In the former case, exemplified by [61], one exploits only the information contained in the input functions of different operators, which typically limits the model's ability to handle highly varying families of operators and prevents generalization to unseen ones. By contrast, in the latter case, methods that employ an operator-encoding strategy [42,44,52,54,58] incorporate an explicit representation of the operator, such as its governing equation, symbolic form, text, or task label, alongside the corresponding input functions. This additional encoding generally yields stronger generalization and represents a potential construction for PDE foundation models. Notably, the inclusion of operator information enables zero-shot generalization to new PDE tasks, as shown in [54], and such approaches have demonstrated promising capabilities for addressing out-of-distribution tasks without costly retraining.

Despite these advances, a rigorous theoretical understanding of the expressivity and scaling behavior of neural networks in multi-operator regimes remains limited. For learning several single operators, our work provides guidance for architectural design to leverage common structure and achieve optimal scaling complexity. For multiple operator learning, our work introduces new architectures and provides the analysis of universal approximation as well as scaling behavior.

Neural Operator Architectures A variety of neural operator architectures have been developed to approximate mappings between infinite-dimensional function spaces efficiently. Among the most widely used are Deep Operator Networks (DeepONets) [45], which rely on low-rank functional decompositions; Fourier Neural Operators [39], motivated by Fourier spectral methods; and Deep Green Networks [6, 16], which learn Green's functions of PDEs directly. These models differ primarily in their structural assumptions which in

turn govern their discretization strategy, scalability, and domain of applicability. Several variants, including Graph Neural Operators and Multipole Graph Neural Operators [1, 38], further leverage sparsity or multiscale interactions to reduce computational cost. We refer to [17, 32] and references therein for further models in operator learning.

The proposed MNO and MONet architectures retain the separable structure characteristic of DeepONet, while extending these models to the multiple operator learning regime.

Expressivity and Scaling Laws The foundation of operator-learning theory rests on universal approximation results, which establish that a given architecture can approximate a broad class of operators to arbitrary accuracy. The development of an operator network and the study of universal approximation for mappings between spaces of scalar-valued functions is due to [11, 12]. Extensions to DeepONet were provided by [34, 41], to the Fourier Neural Operator by [30], and to PCA-Net by [4] etc. Further notable developments related to this work include [8, 9, 22, 25, 31, 62, 64].

Beyond universal approximation, neural scaling laws provide a quantitative framework for characterizing how network performance scales with data size, model capacity, and computational costs. Developing a theoretical foundation for these laws is essential, as it enables rigorous analysis of generalization error in deep learning and offers predictive insight into how performance improves with increasing data, model complexity, or training time [27]. Empirical studies such as [13] explored the cost–accuracy trade-off across neural operator architectures, quantifying how network size and data availability affect approximation error. In [41], the scaling laws and complexity for deep ReLU networks and DeepONet were rigorously derived and analyzed. Additionally, complexity analyses were carried out theoretically for DeepONet by [34] and extended to PCA-Net in [33]. Related analyses can be found in [15, 21, 35, 47]. Sample complexity bounds for DeepONet and related models are studied in [40, 41], and out-of-distribution generalization estimates in [36].

In the context of multiple-operator learning, empirical analyses have recently been reported in [26, 55]. In this work, we establish the universal approximation of MNO and MONet for multiple operators. We also partly extend the work in [41] to the multiple operator setting and derive scaling laws for MNO and related models.

The remainder of the paper is structured as follows: in Section 2, we review the mathematical background relevant to our proposed methods; in Section 3, we present our main results; in Section 4, we provide detailed proofs of our results; in Section 5, we show the strong empirical performance of our proposed models for multiple operator learning; and in Section 6, we conclude with a summary of our contributions and a discussion of potential directions for future work.

2 Background

2.1 Operator learning

In this section, we recall key results in operator learning, related to the framework introduced by [12], which forms the basis for our subsequent universal approximation analysis. We start by defining the class of Tauber–Wiener (TW) activation functions.

Definition 2.1 (Tauber–Wiener functions)**.** *A function $\sigma : \mathbb{R} \to \mathbb{R}$ is called a Tauber–Wiener function if for all $a < b$, $\varepsilon > 0$, and $f \in \mathrm{C}^0([a, b])$, there exists a linear combination $g(x) = \sum_{i=1}^{N} c_i\, \sigma(\lambda_i x + \theta_i)$ such that*

$$\|f - g\|_{\mathrm{C}^0([a,b])} < \varepsilon,$$

where $N = N(\varepsilon)$ depends on the desired accuracy.

The above definition requires that the activation function σ enable the construction of dense subsets of $\mathrm{C}^0([a, b])$. Typical examples of activation functions satisfying this condition, i.e., belonging to the class of Tauber–Wiener functions, include the hyperbolic tangent, bounded sigmoid functions, Gaussian functions, and oscillatory functions such as the sine. In particular, the ReLU activation $\sigma(x) = \max(0, x)$ is also Tauber-Wiener: indeed, the mapping

$$\phi \mapsto \int_{\mathbb{R}} \max(0, x)\, \phi(x)\, dx$$

defines a continuous linear functional on the Schwartz space, and since ReLU is non-polynomial, [12, Theorem 1] ensures that it belongs to the Tauber-Wiener class. ReLU is a popular choice in practice, and we adopt it as our activation function in the experiments presented in Section 5.

Assuming a TW activation function, we introduce the following network, which we denote by Net.

Definition 2.2 (Net Network)**.** *For fixed positive integers m, n, p, constants $c_{ki}, \zeta_k, \theta_{ki}, \xi_{kij} \in \mathbb{R}$, points $\omega_k \in \mathbb{R}^n$, $x_j \in \Omega_U$ ($i = 1, \dots, n$, $k = 1, \dots, p$, $j = 1, \dots, m$), we define a* Net *network as:*

$$\mathrm{Net}[u](x) = \sum_{k=1}^{p} \sum_{i=1}^{n} c_{ki}\, \sigma \left(\sum_{j=1}^{m} \xi_{kij} u(x_j) + \theta_{ki} \right) \cdot \sigma(\omega_k \cdot x + \zeta_k) \tag{1}$$

for a continuous function $u : \Omega_U \mapsto \mathbb{R}$, $x \in \mathbb{R}^n$ and for some activation function $\sigma \in TW$.

Note that the Net network defined in Eq. (1) can also be re-written as

$$\mathrm{Net}[u](x) = \sum_{k=1}^{p} b_k(u) \tau_k(x)$$

with $b_k(u) = \sum_{i=1}^{n} c_{ki} \sigma \left(\sum_{j=1}^{m} \xi_{kij} u(x_j) + \theta_{ki} \right)$ and $\tau_k(x) = \sigma(\omega_k \cdot x + \zeta_k)$ both being shallow networks. If we extend the latter to deep networks, one recovers the popular DeepONet architecture [45]. In this way, the network is a linear combination of the product of nonlinear (separated) sub-networks.

The Net network enjoys a universal approximation property for nonlinear continuous operators over compact sets.

Theorem 2.3 (Universal Approximation Theorem for Single Operator [12])**.** *Suppose that Assumptions **A.1**, **S.2** and **S.3** hold. Let G be a nonlinear continuous operator mapping $U \mapsto V$, then, for any $\varepsilon > 0$, there exists a neural network defined in Eq.* (1)*, such that*

$$\|G[u](x) - \mathrm{Net}[u](x)\|_{\mathrm{L}^\infty(U \times \Omega_V)} < \varepsilon.$$

In Section 3, we will introduce new neural network architectures and extend Theorem 2.3 to the multiple operator setting. To prepare for this, it is helpful to outline the proof strategy of Theorem 2.3 and introduce the technical tools it relies on, which will also be used in the proof of Theorem 3.4. The main idea is to sequentially separate the input variables of the operator G, thereby reducing the operator approximation problem to the task of approximating functions in finite-dimensional spaces. This reduction is supported by the following result, which guarantees that continuous functions can be uniformly approximated by neural networks with TW activations.

Theorem 2.4 (Universal Approximation for Functions [12])**.** *Suppose that Assumption **S.2** holds and let σ be a TW function. Then, for any $\varepsilon > 0$, there exist $N \in \mathbb{N}$, $\theta_i \in \mathbb{R}$, $\omega_i \in \mathbb{R}^n$, and continuous linear functionals $c_i : U \mapsto \mathbb{R}$ such that*

$$\left| f(x) - \sum_{i=1}^{N} c_i(f) \sigma(\omega_i \cdot x + \theta_i) \right| < \varepsilon$$

holds for all $x \in \Omega_U$ and $f \in U$.

Specifically, for a fixed $u \in U$, the mapping $G[u] \in V$ is a function $G[u] : \Omega_V \mapsto \mathbb{R}$, and Theorem 2.4 stipulates the existence of functionals $\{c_i : V \to \mathbb{R}\}_{i=1}^{N}$ such that

$$\left| G[u](x) - \sum_{i=1}^{N} c_i(G[u])\, \sigma(\omega_i \cdot x + \theta_i) \right| < \varepsilon.$$

The next step is to approximate the continuous functionals $c_i : V \to \mathbb{R}$. To this end, one constructs a sequence of finite-dimensional subspaces $V_{\eta_k} \subseteq V$ that approximate V increasingly well: for every $v \in V$ and $\delta > 0$, there exist $k \in \mathbb{N}$ and $v_k \in V_{\eta_k}$ such that $\|v - v_k\| < \delta$. By continuity, it follows that $|c_i(v) - c_i(v_k)| < \varepsilon$. Moreover, on each finite-dimensional subspace V_{η_k}, the functional c_i can be identified with a function

$\hat{c}_i : \mathbb{R}^{\dim(V_{\eta_k})} \to \mathbb{R}$, which can itself be approximated by Theorem 2.4. Denoting this approximation by $N(v_k)$, the triangle inequality yields

$$|c_i(v) - N(v_k)| \leq |c_i(v) - c_i(v_k)| + |\hat{c}_i(v_k) - N(v_k)|,$$

which completes the argument. Below, we describe the construction and approximation properties of the subsets V_k which we also use in the proof of Theorem 3.4.

First, we recall that if V is a compact subset of $\mathrm{C}^0(\Omega_V)$ where Ω_V is itself compact, then it is uniformly bounded and equicontinuous by the Arzelà-Ascoli theorem. Therefore, there is a decreasing sequence $\eta_1 > \eta_2 > \cdots > \eta_n \to 0$ and $\delta_1 > \delta_2 > \cdots > \delta_n \to 0$ such that if $\|x - y\| < \eta_k$, then

$$|v(x) - v(y)| < \delta_k \tag{2}$$

for all $v \in V$. Then, by the compactness of Ω_V and induction, we can find a sequence $\{x_i\}_{i=1}^{\infty} \subseteq \Omega_V$ and a sequence of positive integers $n(\eta_1) < n(\eta_2) < \cdots < n(\eta_k) \to \infty$, such that the first $n(\eta_k)$ elements

$$N(\eta_k) = \{x_1, \cdots, x_{n_{\eta_k}}\} \tag{3}$$

is an η_k-net in Ω_V.

For each η_k-net and index $1 \leq j \leq n(\eta_k)$, we define functions

$$T^*_{\eta_k,j}(x) = \begin{cases} 1 - \frac{\|x - x_j\|}{\eta_k} & \text{if } \|x - x_j\| \leq \eta_k \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \qquad T_{\eta_k,j}(x) = \frac{T^*_{\eta_k,j}(x)}{\sum_{j=1}^{n(\eta_k)} T^*_{\eta_k,j}(x)}.$$

Note that $\{T_{\eta_k,j}(x)\}_{j=1}^{n(\eta_k)}$ is a partition of unity, i.e., $0 \leq T_{\eta_k,j}(x) \leq 1$, $\sum_{j=1}^{n(\eta_k)} T_{\eta_k,j}(x) \equiv 1$, and $T_{\eta_k,j}(x) = 0$ if $\|x - x_j\| > \eta_k$. Furthermore, the functions $T_{\eta_k,j}(x)$ act as basis elements of the finite-dimensional space $V_{\eta_k} = \{v_{\eta_k} : v \in V\}$ where, for each $v \in V$, v_{η_k} is defined as

$$v_{\eta_k}(x) := \sum_{j=1}^{n(\eta_k)} v(x_j) T_{\eta_k,j}(x). \tag{4}$$

Finally, we let $V^* = V \cup (\bigcup_{k=1}^{\infty} V_{\eta_k})$. Equation (4) essentially maps a finite-dimensional encoding of v back into a function v_{η_k}. The approximation properties of v_{η_k} are summarized in the next lemma.

Lemma 2.5 (Finite-dimensional Approximations of Function Spaces [12])**.** *Assume that V is a compact subset of $\mathrm{C}^0(\Omega_V)$ where Ω_V is itself compact.*

1. *For each fixed k, V_{η_k} is a compact set of dimension $n(\eta_k)$ in $C(\Omega_V)$.*
2. *For every $v \in V$, there exists $v_{\eta_k} \in V_{\eta_k}$ with*

$$\|v - v_{\eta_k}\|_{C(\Omega_V)} < \delta_k.$$

3. *V^* is a compact set in $C(\Omega_V)$.*

2.2 Scaling laws for operator learning

In this section, we review the main ideas behind establishing scaling laws for (multiple) operator learning, i.e. obtaining rates of convergence for the approximation of operators using neural networks. In particular, we focus on the setting in [41] which underpins our analysis.

We start by defining the following class of neural networks. This class is both general and flexible, encompassing a wide family of architectures, and can be readily implemented using standard deep learning frameworks.

Definition 2.6 (Feedforward ReLU Network Class)**.** *Let $q : \mathbb{R}^{d_1} \to \mathbb{R}$ be a feedforward ReLU network defined as*

$$q(x) = W_L \cdot \mathrm{ReLU}\left(W_{L-1} \cdots \mathrm{ReLU}(W_1 x + b_1) + \cdots + b_{L-1}\right) + b_L,$$

where W_ℓ are weight matrices, b_ℓ are bias vectors, and $\mathrm{ReLU}(a) = \max\{a, 0\}$ is applied element-wise.

We define the class of such feedforward networks with ReLU activations:

$$\mathcal{F}_{\mathrm{NN}}(d_1, d_2, L, p, K, \kappa, R) = \left\{ [q_1, q_2, \ldots, q_{d_2}]^\top \in \mathbb{R}^{d_2} \;\middle|\; \begin{array}{l} \text{each } q_k : \mathbb{R}^{d_1} \to \mathbb{R} \text{ has the above form with} \\ L \text{ layers, width bounded by } p, \\ \|q_k\|_{\mathrm{L}^\infty} \le R, \quad \|W_\ell\|_{\infty,\infty} \le \kappa, \quad \|b_\ell\|_\infty \le \kappa, \\ \sum_{\ell=1}^{L} \left(\|W_\ell\|_0 + \|b_\ell\|_0\right) \le K \end{array} \right\},$$

where

- $\|q\|_{\mathrm{L}^\infty} = \sup_{x \in \Omega} |q(x)|$,
- $\|W_\ell\|_{\infty,\infty} = \max_{i,j} |[W_\ell]_{ij}|$,
- $\|b_\ell\|_\infty = \max_i |[b_\ell]_i|$,
- $\|\cdot\|_0$ *denotes the number of nonzero elements.*

This network class consists of vector-valued functions with input dimension d_1, output dimension d_2, depth L, width at most p, at most K nonzero parameters, all bounded in magnitude by κ, and uniformly bounded output norm by R.

In analogy with our discussion in Section 2.1, scaling laws are derived sequentially by fixing the inputs of the operator G to be approximated. This requires quantitative approximation results for both functions and functionals using the network class of Definition 2.6, which can then be combined to obtain operator-level guarantees. Specifically, we will use the following result on function approximation in the proofs of Theorems 3.6, 3.8 and 3.16.

Theorem 2.7 (Function Approximation [41])**.** *Let $d_U > 0$ be an integer, $\gamma_1, \beta_U, L_U > 0$ be constants and assume that $U(d_U, \gamma_U, \beta_U, L_U)$ satisfies Assumption **S.4**. There exists some constant C depending on γ_U and L_U such that the following holds. For any $\varepsilon > 0$,*

- *let $N = C\sqrt{d_U}\varepsilon^{-1}$ and let $\{c_k\}_{k=1}^{N^{d_U}}$ be a uniform grid on Ω_U with spacing $2\gamma_U/N$ along each dimension;*
- *consider the network architecture $\mathcal{F}_{\mathrm{NN}}(d_U, 1, L, p, K, \kappa, R)$ with parameters scaling as*

$$\begin{aligned} &L = \mathcal{O}\left(d_U^2 \log d_U + d_U^2 \log(\varepsilon^{-1})\right), \quad && p = \mathcal{O}(1), \quad && K = \mathcal{O}\left(d_U^2 \log d_U + d_U^2 \log(\varepsilon^{-1})\right), \\ &\kappa = \mathcal{O}(d_U^{d_U/2+1} \varepsilon^{-d_U-1}), && && R = 1 \end{aligned}$$

  where the constants hidden in $\mathcal{O}$ depend on γ_U and L_U.

Then, there exists networks $\{q_k\}_{k=1}^{N^{d_U}} \subset \mathcal{F}_{\mathrm{NN}}(d_U, 1, L, p, K, \kappa, R)$ such that

$$\left\| u - \sum_{k=1}^{N^{d_U}} u(c_k) q_k \right\|_{\mathrm{L}^\infty(\Omega_U)} \le \varepsilon.$$

for any $u \in U$.

In Section 3, we present slightly modified versions of the functional and operator scaling laws from [41], which make certain constants in the approximating network explicit and play a central role in the proof of the multiple operator case, Theorem 3.16.

3 Main results

3.1 Notation, Assumptions and Setting

We denote the Lebesgue measure on $\mathbb{R}^n$ by λ and write $|\Omega|$ for the Lebesgue measure of a set Ω. We denote the set of continuous function over a set K as $\mathrm{C}^0(K)$ and the set of continuous maps from U to V as $\mathrm{C}^0(U, V)$. For a vector z and a matrix Z, we denote by $[z]_i$ and $[Z]_{ij}$ their i-th and ij-th element respectively. We denote the ball of radius δ with center x by $\mathcal{B}_\delta(x)$.

3.1.1 Assumptions

Assumptions 1. We make the following assumptions on the activation functions used in the operator networks.

A.1 The activation function σ is a Tauber-Wiener function.

A.2 The activation function σ is continuous and/or bounded.

Assumptions 2. We make the following assumption on our spaces.

S.1 The space of $W \subseteq \mathrm{C}^0(\Omega_W)$ is a compact subspace where Ω_W is a compact subset of the Banach space $\mathcal{A}$.

S.2 The space $U \subseteq \mathrm{C}^0(\Omega_U)$ is a compact subspace where Ω_U is a compact subset of the Banach space $\mathcal{U}$.

S.3 The space V is $\mathrm{C}^0(\Omega_V)$ where Ω_V is a compact subset of $\mathbb{R}^n$.

S.4 The space $U(d_U, \gamma_U, L_U, \beta_U)$ is a function set such that

(a) any function $u \in U$ is defined on $\Omega_U := [-\gamma_U, \gamma_U]^{d_U}$;

(b) for all functions $u \in U$ and $x, y \in \Omega_U$, we have

$$|u(x) - u(y)| \leq L_U |x - y|;$$

(c) for all functions $u \in U$, we have $\|u\|_{\mathrm{L}^\infty} \leq \beta_U$.

Assumptions 3. We make the following assumption on the measures.

M.1 ν is a probability measure on W.

M.2 μ is a probability measure on U.

Assumptions 4. We make the following assumption on the operators.

O.1 For every $\alpha \in W$, the operator $G[\alpha] : U \mapsto V$ is nonlinear and continuous.

O.2 The map $\alpha \in W \mapsto G[\alpha]$ is continuous.

O.3 The map $\alpha \in W \mapsto G[\alpha]$ is Borel measurable and $G[\alpha][u](x) \in \mathrm{L}^2_{\nu \times \mu \times \lambda}(W \times U \times V)$

3.1.2 Multiple Operator Network Architectures

We introduce two neural network architectures for approximating multi-operator mappings. First, we consider the Multiple Operator Network (MONet) which is a direct extension of the neural network in Eq. (1) and we will show that it enjoys universal approximation properties for continuous and measurable multiple operators mappings in Theorems 3.4 and 3.5.

Definition 3.1 (MONet Network)**.** *For fixed positive integers M, N, P, m, p, constants c_{kij}, ζ_k, ξ_{kil}, φ_{kijh}, ρ_{kij}, $\theta_{ki} \in \mathbb{R}$, points $\omega_k \in \mathbb{R}^n$, $x_l \in \Omega_U$, $z_h \in \Omega_W$ ($i = 1, \ldots, M$; $k = 1, \ldots, N$; $j = 1, \ldots, P$; $h = 1, \ldots, p$; $l = 1, \ldots, m$), we define a* MONet *network as:*

$$\text{MONet}[\alpha][u](x) = \sum_{k=1}^{N} \sum_{i=1}^{M} \tau_k(x) b_{ki}(u) L_{ki}(\alpha) = \sum_{k=1}^{N} \tau_k(x) \sum_{i=1}^{M} b_{ki}(u) L_{ki}(\alpha) \tag{5}$$

for continuous functions $\alpha \in W : \Omega_W \mapsto \mathbb{R}$ and $u \in U : \Omega_U \mapsto \mathbb{R}$, $x \in \mathbb{R}^n$, activation function $\sigma \in TW$ and networks $\tau_k(x) = \sigma(\omega_k \cdot x + \zeta_k)$, $b_{ki}(u) = \sigma\left(\sum_{l=1}^{m} \xi_{kil} u(x_l) + \theta_{ki}\right)$ and

$$L_{ki}(\alpha) = \sum_{j=1}^{P} c_{kij} \sigma \left(\sum_{h=1}^{p} \varphi_{kijh} \alpha(z_h) + \rho_{kij} \right).$$

Note that the notation used in the operator networks assumes that the inputs are continuous functions; however, the input functions are encoded as finite dimensional vectors by first being evaluated on some points. In some sense, the resulting encoded vectors are the true inputs to the networks. In all the proofs describing a specific architecture as in the ones of Theorem 3.4, Lemma 4.2, Theorem 3.8 and Theorem 3.16, precise statements will be made.

Remark 3.2 ($\text{MONet}_{\text{vect}}$ Network). The network in Eq. (5) can be simplified in the case where the parameter inputs are finite-dimensional, i.e., $\alpha \in \mathbb{R}^p$. For fixed positive integers M, N, P, m, p, constants c_{kij}, ζ_k, ξ_{kil}, φ_{kijh}, ρ_{kij}, $\theta_{ki} \in \mathbb{R}$, points $\omega_k \in \mathbb{R}^n$, $x_l \in \Omega_U$ ($i = 1, \ldots, M$; $k = 1, \ldots, N$; $j = 1, \ldots, P$; $h = 1, \ldots, p$; $l = 1, \ldots, m$), we define a $\text{MONet}_{\text{vect}}$ network with vector parameter $\alpha \in \mathbb{R}^p$ as

$$\text{MONet}_{\text{vect}}[\alpha][u](x) = \sum_{k=1}^{N} \sum_{i=1}^{M} \sum_{j=1}^{P} c_{kij}\, \sigma \left(\sum_{h=1}^{p} \varphi_{kijh} [\alpha]_h + \rho_{kij} \right) \cdot \sigma \left(\sum_{l=1}^{m} \xi_{kil} u(x_l) + \theta_{ki} \right) \cdot \sigma(\omega_k \cdot x + \zeta_k) \tag{6}$$

for a continuous function $u : \Omega_U \mapsto \mathbb{R}$, point $x \in \mathbb{R}^n$ and some activation function $\sigma \in TW$. The proof of the universal approximation for finite dimensional α is given in Corollary 4.1.

We introduce the Multiple Nonlinear Operator (MNO) Network, which is shown to provide strong empirical results in Section 5. We establish scaling laws in Theorem 3.16 for this architecture.

Definition 3.3 (MNO Network)**.** *For fixed positive integers $P, H^{(p)}$, $1 \leq p \leq P$, we define a* MNO *network as*

$$\text{MNO}[\alpha][u](x) = \sum_{p=1}^{P} l_p(\alpha) \sum_{k=1}^{H^{(p)}} b_{pk}(u) \tau_{pk}(x) = \sum_{p=1}^{P} \sum_{k=1}^{H^{(p)}} l_p(\alpha) b_{pk}(u) \tau_{pk}(x)$$

for continuous functions α, u, and networks l_p, b_k, τ_{pk} in some classes $\mathcal{F}_{\text{NN}}$.

We summarize both network architectures and their associated expressivity guarantees in Table 3.

3.2 Main results

3.2.1 Universal Approximation

In this section, we show that both network architectures introduced in Section 3.1.2 can approximate families of nonlinear operators. The first result is analogous to classical universal approximation results for neural networks. In particular, it assumes that all our functions are continuous. In the theorems below, NN refers to both MNO and MONet.

Theorem 3.4 (Universal Approximation Theorem for Multiple Operators in L^∞)**.** *Assume that Assumptions **A.1**, **S.1**, **S.2**, **S.3**, **O.1** and **O.2** hold. Then for any $\varepsilon > 0$, there exists a network* NN*, of the form given in Definition 3.1 or Definition 3.3 (with l_p, b_{pk}, and τ_{pk} being defined as in Definition 3.1), such that*

$$\|G[\alpha][u](x) - \text{NN}[\boldsymbol{\alpha}][\mathbf{u}](x)\|_{\text{L}^\infty(W \times U \times \Omega_V)} < \varepsilon \tag{7}$$

for all functions $\alpha \in W$ and $u \in U$ and where $\boldsymbol{\alpha}$ and $\mathbf{u}$ are discretizations of α and u that only depend on W and U respectively.

	MONet	MNO
Definition	Definition 3.1	Definition 3.3
Expression	$\sum_{k=1}^{N}\sum_{i=1}^{M}\tau_k(x)\,b_{ki}(u)\,L_{ki}(\alpha)$	$\sum_{p=1}^{P}\sum_{k=1}^{H^{(p)}} l_p(\alpha)\,b_{pk}(u)\,\tau_{pk}(x)$
Components	τ_k, b_{ki}, L_{ki} are shallow networks	l_p, b_{pk}, τ_{pk} are deep networks in $\mathcal{F}_{\mathrm{NN}}$
UAP	Approximates continuous and measurable multiple operator mappings (Theorems 3.4 and 3.5)	Approximates continuous and measurable multiple operator mappings (Theorems 3.4 and 3.5)
Scaling laws	—	Quantitative approximation for Lipschitz multiple operator mappings (Theorem 3.16)

Table 3: Comparison of MNO and MONet architectures: definition, expression, component type, universal approximation properties (UAP), and scaling laws.

The proof is provided in Section 4.1. Next, we relax the continuity requirement on the map $\alpha \mapsto G[\alpha]$, extending the result from continuous to measurable operator families. In this more general setting, the approximation is obtained in the L^2-norm rather than the L^∞-norm.

Theorem 3.5 (Universal Approximation Theorem for Multiple Operators in L^2)**.** *Assume that Assumptions **A.1**, **A.2**, **S.1**, **S.2**, **S.3**, **M.1**, **M.2**, **O.1** and **O.3** hold. Then, for every $\varepsilon > 0$, there exists a network* NN*, of the form given in Definition 3.1 (with $L_{ki}(\alpha) = \gamma_{ki}\left(\sum_{j=1}^{P} c_{kij}\sigma\left(\sum_{h=1}^{p}\varphi_{kijh}\alpha(z_h) + \rho_{kij}\right)\right)$ where γ_{ki} are ReLu neural networks) or Definition 3.3 (with l_p, b_{pk}, and τ_{pk} being defined as in Definition 3.1), such that*

$$\|G[\alpha][u](x) - \mathrm{NN}[\boldsymbol{\alpha}][\boldsymbol{u}](x)\|_{\mathrm{L}^2_{\nu\times\mu\times\lambda}(W\times U\times\Omega_V)} < \varepsilon \tag{8}$$

for any functions $\alpha \in W$ and $u \in U$ and where $\boldsymbol{\alpha}$ and $\mathbf{u}$ are discretizations of α and u that only depend on W and U respectively.

The proof is given in Section 4.2.

3.2.2 Scaling Laws

In this section, we establish the scaling laws for the MNO architecture. Our strategy is indirect: we first carry out the analysis for an equivalent, but more explicit, architecture in Theorem 3.16. The scaling laws for MNO then follow as a corollary through suitable reformulations as detailed in Remark 3.19. Moreover, the results derived for the setting of learning several single operators emerge naturally from this analysis. Our analysis proceeds hierarchically, beginning at the functional level, extending to the approximation of (several) single operators, and then yielding the final general multiple operator learning result.

As discussed in Section 2.2, we start by considering a revised version of [41, Theorem 6] for quantitative functional approximation through neural networks. For the sake of completeness, in Section 4.3.1, we provide a modified proof which explicitly determines the values of some of the constants in the approximating network architecture.

Theorem 3.6 (Functional Approximation)**.** *Let $d_U > 0$ be an integer, $\gamma_U, \beta_U, L_U, L_f > 0$ be constants and assume that $U(d_U, \gamma_U, L_U, \beta_U)$ satisfies Assumption **S.4**. Let $f : \{u : \Omega_U \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \le \beta_U\} \mapsto \mathbb{R}$ be a functional such that*

$$|f(u_1) - f(u_2)| \le L_f\|v_1 - v_2\|_{\mathrm{L}^\infty}$$

for all $u_1, u_2 \in \{u : \Omega_U \,|\, \|u\|_{\mathrm{L}^\infty}(\Omega_U) \le \beta_U\}$.

There exists constants C and C_δ depending on β_U, L_f and L_f, L_U respectively such that the following holds. For any $\varepsilon > 0$,

- *let $\delta = C_\delta \varepsilon$ and let $\{c_m\}_{m=1}^{n_{c_U}} \subset \Omega_U$ be points so that $\{\mathcal{B}_\delta(c_m)\}_{m=1}^{n_{c_U}}$ is a cover of Ω_U for some n_{c_U};*
- *let $H = C\sqrt{n_{c_U}}\varepsilon^{-1}$ and consider the network class $\mathcal{F}_{\mathrm{NN}}(n_{c_U}, 1, L, p, K, \kappa, R)$ with parameters scaling as*

$$L = \mathcal{O}\left(n_{c_U}^2 \log(n_{c_U}) + n_{c_U}^2 \log(\varepsilon^{-1})\right), \quad p = \mathcal{O}(1), \quad K = \mathcal{O}\left(n_{c_U}^2 \log n_{c_U} + n_{c_U}^2 \log(\varepsilon^{-1})\right),$$
$$\kappa = \mathcal{O}(n_{c_U}^{n_{c_U}/2+1} \varepsilon^{-n_{c_U}-1}), \qquad R = 1$$

where the constants hidden in $\mathcal{O}$ depend on β_U and L_f.

Then, there exists networks $\{b_k\}_{k=1}^{H^{n_{c_U}}} \subset \mathcal{F}_{\mathrm{NN}}(n_{c_U}, 1, L, p, K, \kappa, R)$ and functions $\{u_k\}_{k=1}^{H^{n_{c_U}}} \subset \{u : \Omega_U \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \leq \beta_U\}$ such that

$$\sup_{u \in U} \left| f(u) - \sum_{k=1}^{H^{n_{c_U}}} f(u_k) b_k(\boldsymbol{u}) \right| \leq \varepsilon, \tag{9}$$

where $\mathbf{u} = (u(c_1), u(c_2), \ldots, u(c_{n_{c_U}}))^\top$.

Remark 3.7 (Uniform functional approximation). We can extend Theorem 3.6 to a set of functionals

$$\{f^{(j)} : \{u : \Omega_{U^{(j)}} \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \leq \beta_U^{(j)}\} \mapsto \mathbb{R} \,|\, \; |f^{(j)}(u_1) - f^{(j)}(u_2)| \leq L^{(j)} \|u_1 - u_2\|_{\mathrm{L}^\infty}\}_{j \in \mathcal{J}}$$

where $\mathcal{J}$ is a (possibly uncountable) index set. For simplicity, we assume that $d_{U^{(j)}} = d_U$ for all $j \in \mathcal{J}$ and define

$$H^{(j)} = H^{n_{c_{U^{(j)}}}}.$$

Case I: $U^{(j)}$ are distinct In the case of distinct $U^{(j)}$, we apply Theorem 3.6 for every j separately and, for every j, obtain (9). Then, we take the supremum over j and have

$$\sup_{j \in \mathcal{J}} \sup_{u \in U^{(j)}} \left| f^{(j)}(u) - \sum_{k=1}^{H^{(j)}} f^{(j)}(u_k^{(j)}) b_k^{(j)}(\boldsymbol{u}^{(j)}) \right| \leq \varepsilon \tag{10}$$

where $\{b_k^{(j)}\}_{k=1}^{H^{(j)}}$ are networks with $b_k^{(j)} \in \mathcal{F}_{\mathrm{NN}}(n_{c_{U^{(j)}}}, 1, L^{(j)}, p^{(j)}, K^{(j)}, \kappa^{(j)}, R^{(j)})$ for any $1 \leq k \leq H^{(j)}$, $\{u_k^{(j)}\}_{k=1}^{H^{(j)}}$ are functions in $\{u : \Omega_{U^{(j)}} \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \leq \beta_U^{(j)}\}$ and $\mathbf{u}^{(j)} = (u(c_1^{(j)}), u(c_2^{(j)}), \ldots, u(c_{n_{c_{U^{(j)}}}}^{(j)}))^T$. All the constants with superscript j are analogous to the ones defined in the statement of Theorem 3.6 using the appropriated quantities related to $U^{(j)}$.

Case II: $U^{(j)} = U$ If we have $U^{(j)} = U$ for all j, the above simplifies. By inspecting the proof of Theorem 3.6, we note that our functional approximation relies on the function approximation Theorem 2.7. In particular, the idea is to transform our functional $f : \{u : \Omega_U \mapsto \mathbb{R} \;|\, \|u\|_{\mathrm{L}^\infty} \leq \beta_U\} \mapsto \mathbb{R}$ into a Lipschitz function $\hat{f} : [-\beta_U, \beta_U]^{n_{c_U}} \mapsto \mathbb{R}$ contained in some class $V(n_{c_U}, \beta_U, L_f, C_{\hat{f}})$. Then, we obtain the approximation result

$$\sup_{x \in [-\beta_U, \beta_U]^{n_{C_U}}} \left| \hat{f}(x) - \sum_{k=1}^{H^{n_{c_U}}} \hat{f}(s_k) b_k(x) \right| \leq \frac{\varepsilon}{2}$$

where the same networks b_k and points s_k can be chosen for any function in the class V. In particular, the only parameters in the class of functions V and in the second part of the approximation (Eq. (45)) that depend on f are L_f and $C_{\hat{f}}$. Therefore, if we consider a set of functionals

$$\{f^{(j)} : \{u : \Omega_U \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \leq \beta_U\} \mapsto \mathbb{R} \,|\, \; |f^{(j)}(u_1) - f^{(j)}(u_2)| \leq L_j \|u_1 - u_2\|_{\mathrm{L}^\infty}\}_{j \in \mathcal{J}},$$

the same argument can be repeated by replacing L_f by $\sup_{j\in\mathcal{J}} L_j$ and $C_{\hat{f}}$ by $\sup_{j\in\mathcal{J}} C_{\hat{f}_j}$: in fact, $\hat{f}^{(j)} \in V(n_{c_U}, \beta_U, \sup_{j\in\mathcal{J}} L_j, \sup_{j\in\mathcal{J}} C_{\hat{f}_j})$. We can conclude that

$$\sup_{j\in\mathcal{J}} \sup_{u\in U} \left| f^{(j)}(u) - \sum_{k=1}^{H^{n_{c_U}}} f^{(j)}(u_k) b_k(\boldsymbol{u}) \right| \leq \varepsilon. \tag{11}$$

This result will only affect the choice of the constants in the statement of the theorem, none of the scalings. Obviously, this presupposes that $\sup_{j\in\mathcal{J}} L_j < \infty$ and $\sup_{j\in\mathcal{J}} C_{\hat{f}_j} < \infty$. In particular, we note that we can set $\sup_{j\in\mathcal{J}} C_{\hat{f}_j} = \sup_{j\in\mathcal{J}} \sup_{u\in\{u:\Omega_U\mapsto\mathbb{R}\,|\,\|u\|_{\mathrm{L}^\infty}\leq\beta_U^{(j)}\}} f^{(j)}(u)$.

We also note that the case where some $U^{(j)}$ are distinct and some coincide is dealt with similarly, as a combination of Eqs. (10) and (11).

With the explicit functional approximation provided by Theorem 3.6, we now establish a version of the operator scaling laws that includes explicit coefficients for the approximating network. The proof of the theorem is analogous to [41, Theorem 1], just substituting Theorem 3.6 for [41, Theorem 6]. We recall the main steps in Remark 3.9 and prove a very similar statement in Remark 3.13.

Theorem 3.8 (Single Operator Scaling Laws)**.** *Let $d_U, d_V > 0$ be integers, $\gamma_U, \gamma_V, \beta_U, \beta_V, L_U, L_V, L_G > 0$, and assume that $U(d_U, \gamma_U, L_U, \beta_U)$ and $V(d_V, \gamma_V, L_V, \beta_V)$ satisfy Assumption **S.4**. Let G be an operator such that $G : \{u : \Omega_U \mapsto \mathbb{R}\,|\,\|u\|_{\mathrm{L}^\infty} \leq \beta_U\} \mapsto V$. Furthermore, assume that G satisfies*

$$\|G(u_1) - G(u_2)\|_{\mathrm{L}^\infty(\Omega_V)} \leq L_G \|u_1 - u_2\|_{\mathrm{L}^r(\Omega_U)} \tag{12}$$

for some $r \geq 1$ and for any $u_1, u_2 \in \{u : \Omega_U \mapsto \mathbb{R}\,|\,\|u\|_{\mathrm{L}^\infty} \leq \beta_U\}$.

There exists constants C depending on γ_V, L_V, C_δ depending on $L_G, d_U, \gamma_U, r, L_U$ and C' depending on $\beta_U, L_G, d_U, \gamma_U, r$ such that the following holds. For any $\varepsilon > 0$,

- *let $N = 2C\sqrt{d_V}\varepsilon^{-1}$ and consider the network class $\mathcal{F}_1 = \mathcal{F}_{\mathrm{NN}}(d_V, 1, L_1, p_1, K_1, \kappa_1, R_1)$ with parameters scaling as*

$$L_1 = \mathcal{O}\left(d_V^2 \log d_V + d_V^2 \log(\varepsilon^{-1})\right), \quad p_1 = \mathcal{O}(1), \quad K_1 = \mathcal{O}\left(d_V^2 \log d_V + d_V^2 \log(\varepsilon^{-1})\right),$$
$$\kappa_1 = \mathcal{O}(d_V^{d_V/2+1}\varepsilon^{-d_V-1}), \qquad R_1 = 1.$$

  where the constants hidden in $\mathcal{O}$ depend on γ_V and L_V;

- *let $\{v_\ell\}_{\ell=1}^{N^{d_V}} \subset \Omega_V$ be a uniform grid with spacing $2\gamma_V/N$ along each dimension;*

- *let $\delta = \frac{C_\delta \varepsilon^{1+d_V}}{2^{d_V+1}(C\sqrt{d_V})^{d_V}}$ and let $\{c_m\}_{m=1}^{n_{c_U}} \subset \Omega_U$ be points so that $\{\mathcal{B}_\delta(c_m)\}_{m=1}^{n_{c_U}}$ is a cover of Ω_U for some n_{c_U};*

- *let $H = 2^{d_V+1}C'\sqrt{n_{c_U}}(C\sqrt{d_V})^{d_V}\varepsilon^{-(d_V+1)}$ and consider the network class $\mathcal{F}_2 = \mathcal{F}_{\mathrm{NN}}(n_{c_U}, 1, L_2, p_2, K_2, \kappa_2, R_2)$ with parameters scaling as*

$$L_2 = \mathcal{O}\left(n_{c_U}^2 \log n_{c_U} + n_{c_U}^2 (d_V+1)\log(\varepsilon^{-1}) + n_{c_U}^2 \log(2^{d_V+1}(C\sqrt{d_V})^{d_V})\right), \quad p_2 = \mathcal{O}(1),$$
$$K_2 = \mathcal{O}\left(n_{c_U}^2 \log n_{c_U} + n_{c_U}^2 (d_V+1)\log(\varepsilon^{-1}) + n_{c_U}^2 \log(2^{d_V+1}(C\sqrt{d_V})^{d_V})\right),$$
$$\kappa_2 = \mathcal{O}(n_{c_U}^{n_{c_U}/2+1}\varepsilon^{-(d_V+1)(n_{c_U}+1)}[2^{d_V+1}(C\sqrt{d_V})^{d_V}]^{n_{c_U}+1}), \qquad R_2 = 1$$

  where the constants hidden in $\mathcal{O}$ depend on $\beta_U, L_G, d_U, \gamma_U, r$.

Then, there exists networks $\{\tau_\ell\}_{\ell=1}^{N^{d_V}} \subset \mathcal{F}_1$, networks $\{b_k\}_{k=1}^{H^{n_{c_U}}} \subset \mathcal{F}_2$ and functions $\{u_k\}_{k=1}^{H^{n_{c_U}}} \subset \{u : \Omega_U \mapsto \mathbb{R}\,|\,\|u\|_{\mathrm{L}^\infty} \leq \beta_U\}$ such that

$$\sup_{u\in U} \sup_{x\in\Omega_V} \left| G[u](x) - \sum_{\ell=1}^{N^{d_V}} \sum_{k=1}^{H^{n_{c_U}}} G[u_k](v_\ell) b_k(\mathbf{u}) \tau_\ell(x) \right| \leq \varepsilon, \tag{13}$$

where $\mathbf{u} = (u(c_1), u(c_2), ..., u(c_{n_{c_U}}))^\top$ is a discretization of u.

Remark 3.9 (Uniform operator approximation). Similarly to Remark 3.7, we extend Theorem 3.8 to a set of operators

$$\{G^{(j)} : \{u : \Omega_{U^{(j)}} \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \leq \beta_{U^{(j)}}\} \mapsto V^{(j)} \,|\, \|G^{(j)}(u_1) - G^{(j)}(u_2)\|_{\mathrm{L}^\infty} \leq L_{G^{(j)}} \|u_1 - u_2\|_{\mathrm{L}^{r^{(j)}}}\}_{j\in\mathcal{J}}$$

where $\mathcal{J}$ is a (possibly uncountable) index set. For simplicity, we assume that $d_{U^{(j)}} = d_U$ and $d_{V^{(j)}} = d_V$ for all $j \in \mathcal{J}$ and define

$$H^{(j)} = H^{n_{c_{U^{(j)}}}}.$$

Case I: $U^{(j)}$ are distinct and $V^{(j)}$ are distinct If the $U^{(j)}$ and $V^{(j)}$ are distinct, we apply Theorem 3.8 first for each $j \in \mathcal{J}$ separately and then take the supremum over all j to obtain:

$$\sup_{j\in\mathcal{J}} \sup_{u\in U^{(j)}} \sup_{x\in\Omega_{V^{(j)}}} \left| G^{(j)}[u](x) - \sum_{\ell=1}^{(N^{(j)})^{d_V}} \sum_{k=1}^{H^{(j)}} G^{(j)}[u_k^{(j)}](v_\ell^{(\ell)}) b_k^{(j)}(\mathbf{u}^{(j)}) \tau_\ell^{(j)}(x) \right| \leq \varepsilon.$$

For the rest of the cases, we need to recall the proof of Theorem 3.8. Specifically, the idea is first to consider, for $u \in U$, the functions $G[u] : \Omega_V \mapsto \mathbb{R}$. The latter are all contained in V and we can therefore apply the function approximation Theorem 2.7 to deduce that for all $u \in U$,

$$\sup_{x\in\Omega_V} \left| G[u](x) - \sum_{\ell=1}^{N^{d_V}} G[u](v_\ell)\tau_\ell(x) \right| \leq \frac{\varepsilon}{2}.$$

Then, we define the functionals $f_\ell(u) = G[u](v_\ell)$ and verify that they are L^∞-Lipschitz on $\{u : \Omega_U \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \leq \beta_U\}$ with Lipschitz constant $L_G|\Omega_V|^{1/r}$. As explained in Remark 3.7, the proof corresponds to the setting where we have N^{d_V} functionals all defined for the same set of functions U, we can apply the formula in Eq. (11) to obtain that, for all $1 \leq \ell \leq N^{d_V}$,

$$\sup_{u\in U} \left| f_\ell(u) - \sum_{k=1}^{H^{n_{c_U}}} f_\ell(u_k) b_k(\mathbf{u}) \right| = \sup_{u\in U} \left| G[u](v_\ell) - \sum_{k=1}^{H^{n_{c_U}}} G[u_k](v_\ell) b_k(\mathbf{u}) \right| \leq \varepsilon_0.$$

Combining both estimates, we conclude with

$$\begin{aligned}
&\sup_{u\in U} \sup_{x\in\Omega_V} \left| G[u](x) - \sum_{\ell=1}^{N^{d_V}} \sum_{k=1}^{H^{n_{c_U}}} G[u_k](v_\ell) b_k(\mathbf{u}^{(j)}) \tau_\ell(x) \right| \\
&\leq \sup_{u\in U} \sup_{x\in\Omega_V} \left| G[u](x) - \sum_{\ell=1}^{N^{d_V}} G[u](v_\ell)\tau_\ell(x) \right| + \sup_{u\in U} \sup_{x\in\Omega_V} \sum_{\ell=1}^{N^{d_V}} |\tau_\ell(x)| \left| G[u](v_\ell) - \sum_{k=1}^{H^{n_{c_U}}} G[u_k](v_\ell) b_k(\mathbf{u}^{(j)}) \right| \\
&\leq \frac{\varepsilon}{2} + \varepsilon_0 N^{d_V}.
\end{aligned}$$

By picking $\varepsilon_0 = \varepsilon/(2N^{d_V}) = \mathcal{O}(\varepsilon^{d_V+1})$, we obtain the result.

Case II: $U^{(j)} = U$ and $V^{(j)}$ are distinct Let us now assume that $U^{(j)} = U$. The first step of the proof can be repeated for every j separately to obtain that for every $u \in U$,

$$\sup_{j\in J} \sup_{x\in\Omega_{V^{(j)}}} \left| G^{(j)}[u](x) - \sum_{\ell=1}^{(N^{(j)})^{d_V}} G[u](v_\ell^{(j)}) \tau_\ell^{(j)}(x) \right| \leq \frac{\varepsilon}{2}.$$

Next, we can define the functionals

$$f_\ell^{(j)}(u) = G^{(j)}[u](v_\ell^{(j)})$$

and the latter are L^∞-Lipschitz in $\{u : \Omega_U \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \leq \beta_U\}$ with Lipschitz constant $\sup_{j\in\mathcal{J}} |\Omega_U|^{1/r^{(j)}} L_{G^{(j)}}$ if we assume that the latter is finite. If we further assume that

$$\sup_{j\in\mathcal{J}} \sup_{1\leq\ell\leq N^{d_V(j)}} f_\ell^{(j)}(u) = \sup_{j\in\mathcal{J}} \sup_{1\leq\ell\leq N^{d_V(j)}} G^{(j)}[u](v_\ell^{(j)}) \leq \sup_{j\in\mathcal{J}} \sup_{v\in V^{(j)}} \|v\|_{\mathrm{L}^\infty(V^{(j)})} \leq \sup_{j\in\mathcal{J}} \beta_{V^{(j)}} < \infty$$

then, the functionals satisfy all the assumptions in Eq. (11) and we obtain

$$\begin{aligned}\sup_{j\in\mathcal{J}} \sup_{u\in U} \sup_{1\leq\ell\leq(N^{(j)})^{d_V}} \left| f_\ell^{(j)}(u) - \sum_{k=1}^{H^{n_{c_U}}} f_\ell^{(j)}(u_k) b_k(\boldsymbol{u}) \right| &= \sup_{j\in\mathcal{J}} \sup_{u\in U} \left| G^{(j)}[u](x) - \sum_{k=1}^{H^{n_{c_U}}} G^{(j)}[u_k](x) b_k(\boldsymbol{u}) \right| \\ &\leq \frac{\varepsilon}{2\sup_{j\in\mathcal{J}}(N^{(j)})^{d_V}} =: \varepsilon_0.\end{aligned}$$

This also requires that $\sup_{j\in\mathcal{J}}(N^{(j)})^{d_V} < \infty$ and we note a subtle point: in this setting, b_k, u_k and $\mathbf{u}$ can be chosen independently of j. The is possible since the latter are a function of ε_0 which is set to $\frac{\varepsilon}{2\sup_{j\in\mathcal{J}}(N^{(j)})^{d_V}}$, i.e. independent of j, and not to $\frac{\varepsilon}{2(N^{(j)})^{d_V}}$, in which case they would both become dependent of j again. By concluding as above, we obtain:

$$\sup_{j\in\mathcal{J}} \sup_{u\in U} \sup_{x\in\Omega_{V^{(j)}}} \left| G^{(j)}[u](x) - \sum_{\ell=1}^{(N^{(j)})^{d_V}} \sum_{k=1}^{H^{n_{c_U}}} G^{(j)}[u_k](v_\ell^{(j)}) b_k(\mathbf{u}) \tau_\ell^{(j)}(x) \right| \leq \varepsilon.$$

Case III: $U^{(j)}$ are distinct and $V^{(j)} = V$ Next, we assume that $V_j = V$. Since for all $j \in \mathcal{J}$ and $u^{(j)} \in U^{(j)}$, we obtain that $G^{(j)}[u^{(j)}] \in V$, by repeating the first step of the proof, we can choose τ_ℓ and v_ℓ to be independent of j and obtain

$$\sup_{j\in J} \sup_{u^{(j)}\in U^{(j)}} \sup_{x\in\Omega_V} \left| G^{(j)}[u^{(j)}](x) - \sum_{\ell=1}^{N^{d_V}} G^{(j)}[u^{(j)}](v_\ell) \tau_\ell(x) \right| \leq \frac{\varepsilon}{2}.$$

We then define the functionals $f_\ell^{(j)} : \{u^{(j)} : \Omega_{U^{(j)}} \mapsto \mathbb{R} \,|\, \|u^{(j)}\|_{\mathrm{L}^\infty} \leq \beta_{U^{(j)}}\}$ as $f_\ell^{(j)}(u^{(j)}) = G^{(j)}[u^{(j)}](v_\ell)$ and verify that they are L^∞-Lipschitz with Lipschitz constant $|\Omega_{U^{(j)}}|^{1/r^{(j)}} L_G^{(j)}$. We then apply Eq. (10) and obtain that

$$\begin{aligned}&\sup_{j\in\mathcal{J}} \sup_{u^{(j)}\in U^{(j)}} \sup_{1\leq\ell\leq N^{d_V}} \left| f_\ell^{(j)}(u^{(j)}) - \sum_{k=1}^{H^{(j)}} f_\ell^{(j)}(u_k^{(j)}) b_k^{(j)}(\boldsymbol{u}^{(j)}) \right| \\ &= \sup_{j\in\mathcal{J}} \sup_{u^{(j)}\in U^{(j)}} \sup_{1\leq\ell\leq N^{d_V}} \left| G^{(j)}[u^{(j)}](v_\ell) - \sum_{k=1}^{H^{(j)}} G^{(j)}[u_k^{(j)}](v_\ell) b_k^{(j)}(\boldsymbol{u}^{(j)}) \right| \\ &\leq \frac{\varepsilon}{2N^{d_V}}.\end{aligned}$$

Combining both estimates, we conclude that

$$\sup_{j\in J} \sup_{u^{(j)}\in U^{(j)}} \sup_{x\in\Omega_V} \left| G^{(j)}[u^{(j)}](x) - \sum_{\ell=1}^{N^{d_V}} \sum_{k=1}^{H^{(j)}} G^{(j)}[u_k^{(j)}](v_\ell) b_k^{(j)}(\boldsymbol{u}^{(j)}) \tau_\ell(x) \right| \leq \varepsilon.$$

Case IV: $U^{(j)} = U$ and $V^{(j)} = V$ Finally, if $U_j = U$ and $V_j = V$, by combining both of the above, we obtain:

$$\sup_{1\leq j\leq J} \sup_{u\in U} \sup_{x\in\Omega_V} \left| G^{(j)}[u](x) - \sum_{\ell=1}^{N^{d_V}} \sum_{k=1}^{H^{n_{c_U}}} G^{(j)}[u_k](v_\ell) b_k(\mathbf{u}) \tau_\ell(x) \right| \leq \varepsilon. \tag{14}$$

Similarly, we can deal with the case when some $U^{(j)}$ and $V^{(j)}$ are distinct, while some are equal.

Remark 3.10 (Uncountable index set assumptions). Remarks 3.7 and 3.9 have been formulated for uncountably many functionals and operators respectively. While these results are of interest on their own, they also require several assumptions on the finiteness of various constants. In practice, we will apply Remark 3.9 for finitely many operators: this significantly simplifies the necessary assumptions.

Remark 3.11 (Alternative network for the operator approximation). The network appearing in Eq. (13) can be re-written in a slightly different manner. Specifically, we can define

(15)
$$\sum_{\ell=1}^{N^{d_V}} \sum_{k=1}^{H^{n_{c_U}}} G[u_k](v_\ell) b_k(\mathbf{u}) \tau_\ell(x) =: \sum_{k=1}^{H^{n_{c_U}}} b_k(\mathbf{u}) \hat{\tau}_k(x) \text{ or } \sum_{\ell=1}^{N^{d_V}} \sum_{k=1}^{H^{n_{c_U}}} G[u_k](v_\ell) b_k(\mathbf{u}) \tau_\ell(x) =: \sum_{\ell=1}^{N^{d_V}} \hat{b}_\ell(\mathbf{u}) \tau_\ell(x)$$

The networks $\hat{\tau}_k$ and $\hat{b}_\ell$ are in the classes $\mathcal{S}_{N^{d_V}} \mathcal{F}_1$ and $\mathcal{S}_{H^{n_{c_U}}} \mathcal{F}_2$ where $\mathcal{F}_1$ and $\mathcal{F}_2$ are defined in Theorem 3.8 and $\mathcal{S}_j \mathcal{F}$ denotes functions that are linear combinations of j functions in the class $\mathcal{F}$. We note that this is the convention chosen in [41]. These formulations are particularly well-suited for practical applications, as they replace the double summation in Eq. (13) with a single inner product, thereby significantly simplifying implementation.

We also want to consider a set of operators

$$\{G^{(j)} : \{u : \Omega_{U^{(j)}} \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \leq \beta_{U^{(j)}}\} \mapsto V^{(j)} \,|\, \|G^{(j)}(u_1) - G^{(j)}(u_2)\|_{\mathrm{L}^\infty} \leq L_G^{(j)} \|u_1 - u_2\|_{\mathrm{L}^{r^{(j)}}}\}_{j \in \mathcal{J}}.$$

as in Remark 3.9. We distinguish the same following four cases for the alternative network formulations of Eq. (15) which approximate all $G^{(j)}$. In particular, using the formulas derived in Remark 3.9, we obtain:

1. **Case I: $U^{(j)}$ are distinct and $V^{(j)}$ are distinct**
$$\sum_{k=1}^{H^{(j)}} b_k^{(j)}(\mathbf{u}^{(j)}) \hat{\tau}_k^{(j)}(x) \quad \text{or} \quad \sum_{\ell=1}^{(N^{(j)})^{d_V}} \hat{b}_\ell^{(j)}(\mathbf{u}^{(j)}) \tau_\ell^{(j)}(x);$$

2. **Case II: $U^{(j)} = U$ and $V^{(j)}$ are distinct**
$$\sum_{k=1}^{H^{n_{c_U}}} b_k(\mathbf{u}) \hat{\tau}_k^{(j)}(x) \quad \text{or} \quad \sum_{\ell=1}^{(N^{(j)})^{d_V}} \hat{b}_\ell^{(j)}(\mathbf{u}^{(j)}) \tau_\ell^{(j)}(x);$$

3. **Case III: $U^{(j)}$ are distinct and $V^{(j)} = V$**
$$\sum_{k=1}^{H^{(j)}} b_k^{(j)}(\mathbf{u}^{(j)}) \hat{\tau}_k^{(j)}(x) \quad \text{or} \quad \sum_{\ell=1}^{N^{d_V}} \hat{b}_\ell^{(j)}(\mathbf{u}^{(j)}) \tau_\ell(x);$$

4. **Case IV: $U^{(j)} = U$ and $V^{(j)} = V$**
$$\sum_{k=1}^{H^{n_{c_U}}} b_k(\mathbf{u}) \hat{\tau}_k^{(j)}(x) \quad \text{or} \quad \sum_{\ell=1}^{N^{d_V}} \hat{b}_\ell^{(j)}(\mathbf{u}) \tau_\ell(x).$$

The above formulas provide a principled basis for selecting architectures when approximating several single operators simultaneously. In particular, depending on the structure of $U^{(j)}$ and $V^{(j)}$, certain architectures allow the dependence on j to be absorbed into a single network component rather than appearing in multiple components simultaneously. This result also provides a unified framework that encompasses recent approaches such as D2NO [63] and MODNO [61].

Remark 3.12 (Learning several single operator versus multiple operator learning). In practice, one often encounters settings where several distinct operators must be learned (as in Remarks 3.9 and 3.11), either independently or with partial weight sharing—for instance, learning solution maps corresponding to different physical regimes or boundary conditions. At first glance, this may seem equivalent to learning a single parameterized operator, i.e. multiple operator learning.

While both settings involve learning mappings between function spaces, they differ fundamentally in how the dependence on the index variable is treated. In the several single-operator setting, one considers an indexed family of operators $\{G^{(j)} : U^{(j)} \to V^{(j)}\}_{j\in J}$, where J may be finite or uncountable, but the index j does not explicitly enter the learning process. Each operator is represented or trained separately, and any shared structure across j is imposed manually: Remark 3.11 guides the architecture choice in this regard. In contrast, multiple operator learning aims to approximate a single parameterized mapping $G : W \to \{U^{(\alpha)} \to V^{(\alpha)}\}$, where $\alpha \in W$ directly enters the model as an input. This formulation inherently captures how operators vary with α, allowing a single network to interpolate across the entire parameter space rather than fitting a collection of independent mappings. We summarize the main comparison points in Table 4.

	Several Single Operators	Multiple Operator Learning
Formulation	$\{G^{(j)} : U^{(j)} \to V^{(j)}\}_{j\in J}$	$G : W \to \{U^{(\alpha)} \to V^{(\alpha)}\}_{\alpha\in W}$
Dependence on parameter/index	Dependence on j is external to the model	Parameter α is an explicit input to the network
Coupling between operators	Optional, via shared structure of the architecture	Intrinsic, through a single network
Generalization capability	Limited to operators seen during training	Enables interpolation and extrapolation across $\alpha \in W$
Practical interpretation	Independent or weakly coupled tasks	Unified model for a continuum of related tasks

Table 4: Comparison between the settings of several single operators and multiple operator learning. The latter treats the parameter (or index) as an explicit input, enabling a single network to represent a continuously parameterized family of operators.

Remark 3.13 (Balancing functional and spatial scaling complexity in the approximating architecture). Theorem 3.8 establishes specific scaling relations for the space-approximation networks τ_ℓ and the function-approximation networks b_k. These scalings arise naturally from the order of approximation adopted in the proof, namely, approximating functions first and functionals second as recalled in Remark 3.9. If the order is reversed, the resulting derivation yields a different scaling behavior, illustrating that the overall approximation complexity can be redistributed between the two components of the network. This observation highlights a fundamental flexibility in the design of operator-learning architectures: the computational burden can be shifted from one subnetwork to another without altering the expressive power of the overall model.

More precisely, when the order of approximation is inverted, the proof follows the steps outlined below. For $x \in \Omega_V$, we start by defining the functional $f_x : \{u : \Omega_U \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \le \beta_U\} \mapsto \mathbb{R}$ as

$$f_x(u) = G[u](x).$$

In particular, we have that

$$\begin{aligned} |f_x(u_1) - f_x(u_2)| &= |G[u_1](x) - G[u_2](x)| \\ &\le L_G \|u_1 - u_2\|_{\mathrm{L}^r(\Omega_U)} \\ &\le L_G |\Omega_U|^{1/r} \|u_1 - u_2\|_{\mathrm{L}^\infty(\Omega_U)} \end{aligned} \tag{16}$$

where we use (12) for (16). Therefore, we can apply Theorem 3.6. Specifically, for any $\varepsilon_0 > 0$, there exists constants C' and C_δ depending on $\beta_U, L_G|\Omega_U|^{1/r}$ and $L_G|\Omega_U|^{1/r}, L_U$ respectively such that the following holds. There exists

- a constant $\delta := C_\delta \varepsilon_0$ and points $\{c_m\}_{m=1}^{n_{c_U}} \subset \Omega_U$ so that $\{\mathcal{B}_\delta(c_m)\}_{m=1}^{n_{c_U}}$ is a cover of Ω_U for some n_{c_U},
- a network class $\mathcal{F}_2 = \mathcal{F}_{\mathrm{NN}}(n_{c_U}, 1, L_2, p_2, K_2, \kappa_2, R_2)$ whose parameters scale as

$$L_2 = \mathcal{O}\left(n_{c_U}^2 \log(n_{c_U}) + n_{c_U}^2 \log(\varepsilon_0^{-1})\right), \quad p_2 = \mathcal{O}(1), \quad K_2 = \mathcal{O}\left(n_{c_U}^2 \log n_{c_U} + n_{c_U}^2 \log(\varepsilon_0^{-1})\right),$$

$$\kappa_2 = \mathcal{O}(n_{c_U}^{n_{c_U}/2+1}\varepsilon_0^{-n_{c_U}-1}), \qquad R_2 = 1$$

where the constants hidden in $\mathcal{O}$ depend on β_U and $L_G|\Omega_U|^{1/r}$,

- networks $\{b_k\}_{k=1}^{H^{n_{c_U}}} \subset \mathcal{F}_2$ with $H := C'\sqrt{n_{c_U}}\varepsilon_0^{-1}$ and
- functions $\{u_k\}_{k=1}^{H^{n_{c_U}}} \subset \{u : \Omega_U \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \leq \beta_U\}$

such that

$$\sup_{u\in U}\left|f_x(u) - \sum_{k=1}^{H^{n_{c_U}}} f_x(u_k)b_k\left(P_{\mathcal{C}_U}(u)\right)\right| = \sup_{u\in U}\left|G[u](x) - \sum_{k=1}^{H^{n_{c_U}}} G[u_k](x)b_k\left(P_{\mathcal{C}_U}(u)\right)\right| \leq \varepsilon_0. \tag{17}$$

where $P_{\mathcal{C}_U}(u)$ is defined in the proof of Theorem 3.6.

By assumption, $G[u_k] \in V$ for all $1 \leq k \leq H^{n_{c_U}}$ and we can apply Theorem 2.7 to approximate all these functions simultaneously. Specifically, for any $\varepsilon_1 > 0$, there exists a constant C depending on γ_V and L_V such that the following holds. There exists

- a constant $N = C\sqrt{d_V}\varepsilon_1^{-1}$ and points $\{c_k\}_{k=1}^{N^{d_V}}$ which form a uniform grid on Ω_V with spacing $2\gamma_V/N$ along each dimension,
- a network class $\mathcal{F}_1 = \mathcal{F}_{\mathrm{NN}}(d_V, 1, L_1, p_1, K_1, \kappa_1, R_1)$ whose parameters scale as

$$L_1 = \mathcal{O}\left(d_V^2\log d_V + d_V^2\log(\varepsilon_1^{-1})\right), \quad p_1 = \mathcal{O}(1), \quad K_1 = \mathcal{O}\left(d_V^2\log d_V + d_V^2\log(\varepsilon_1^{-1})\right),$$
$$\kappa_1 = \mathcal{O}(d_V^{d_V/2+1}\varepsilon_1^{-d_V-1}), \qquad R_1 = 1$$

where the constants hidden in $\mathcal{O}$ depend on γ_V, L_V and

- networks $\{\tau_\ell\}_{\ell=1}^{N^{d_V}} \subset \mathcal{F}_2$

such that, for every $1 \leq k \leq N^{d_V}$:

$$\sup_{x\in\Omega_V}\left|G[u_k](x) - \sum_{\ell=1}^{N^{d_V}} G[u_k](v_\ell)\tau_\ell(x)\right| \leq \varepsilon_1. \tag{18}$$

Combining both of our bounds Eqs. (17) and (18), we obtain:

$$\begin{aligned}
&\sup_{x\in\Omega_V,\,u\in U}\left|G[u](x) - \sum_{k=1}^{H^{n_{c_U}}}\sum_{\ell=1}^{N^{d_V}} G[u_k](v_\ell)b_k(P_{\mathcal{C}_U}(u))\tau_\ell(x)\right| \\
&\leq \sup_{x\in\Omega_V,\,u\in U}\left|G[u](x) - \sum_{k=1}^{H^{n_{c_U}}} G[u_k](x)b_k(P_{\mathcal{C}_U}(u))\right| \\
&+ \sup_{x\in\Omega_V,\,u\in U}\sum_{k=1}^{H^{n_{c_U}}}\left|G[u_k](x) - \sum_{\ell=1}^{N^{d_V}} G[u_k](v_\ell)\tau_\ell(x)\right||b_k(P_{\mathcal{C}_U}(u))| \\
&\leq \varepsilon_0 + \sup_{x\in\Omega_V,\,u\in U}\sum_{k=1}^{H^{n_{c_U}}}\left|G[u_k](x) - \sum_{\ell=1}^{N^{d_V}} G[u_k](v_\ell)\tau_\ell(x)\right| \qquad (19) \\
&\leq \varepsilon_0 + H^{n_{c_U}}\varepsilon_1. \qquad (20)
\end{aligned}$$

where we use (17) and the fact that $|b_k(P_{\mathcal{C}_U}(u))| \leq 1$ in (19) and (18) for (20).

We conclude by picking $\varepsilon_0 = \varepsilon/2$ and $\varepsilon_1 = \varepsilon/(2H^{n_{c_U}})$. Since $\varepsilon_1 = \varepsilon^{n_{c_U}+1}(C'\sqrt{n_{c_U}})^{-n_{c_U}}2^{-(1+n_{c_U})}$, the final scalings for $\mathcal{F}_1$ are

$$L_1 = \mathcal{O}\left(d_V^2\log d_V + d_V^2\log(\varepsilon^{-(n_{c_U}+1)}) + d_V^2\log(2^{n_{c_U}+1}(C'\sqrt{n_{c_U}})^{n_{c_U}})\right), \quad p_1 = \mathcal{O}(1),$$

$$K_1 = \mathcal{O}\left(d_V^2 \log d_V + d_V^2 \log(\varepsilon^{-(n_{c_U}+1)}) + d_V^2 \log(2^{n_{c_U}+1}(C'\sqrt{n_{c_U}})^{n_{c_U}})\right),$$
$$\kappa_1 = \mathcal{O}(d_V^{d_V/2+1}\varepsilon^{-(d_V+1)(n_{c_U}+1)}\left[2^{n_{c_U}+1}(C'\sqrt{n_{c_U}})^{n_{c_U}}\right]^{-1}), \qquad R_1 = 1,$$
$$N = 2^{n_{c_U}+1}C\sqrt{d_V}(C'\sqrt{n_{c_U}})^{n_{c_U}}\varepsilon^{-(1+n_{c_U})}).$$

We summarize the differences in scaling for the various networks in Table 5. Reversing the approximation order shifts the computational cost: in the original formulation, most complexity lies in the function-approximation networks, whereas in the reversed case, it is transferred to the space-approximation networks.

Remark 3.14 (Total number of parameters for operator learning). We now express the approximation error of the network in Eq. (13) as a function of the total number of parameters, $N_\# := N^{d_V}K_1 + H^{n_{c_U}}K_2$. We note that $n_{c_U} = \mathcal{O}(\varepsilon^{-(1+d_V)d_U})$, by [41, Lemma 2]. We compute as follows:

$$\begin{aligned}
N_\# &= N^{d_V}K_1 + H^{n_{c_U}}K_2 \\
&\asymp \varepsilon^{-d_V}\log(\varepsilon^{-1}) + \left[\sqrt{n_{c_U}}\varepsilon^{-(1+d_V)}\right]^{n_{c_U}}\left(n_{c_U}^2\log n_{c_U} + n_{c_U}^2(d_V+1)\log(\varepsilon^{-1})\right) \\
&\asymp \varepsilon^{-d_V}\log(\varepsilon^{-1}) + \varepsilon^{-[\frac{(1+d_V)d_U}{2}+(1+d_V)]\varepsilon^{-(1+d_V)d_U}}\varepsilon^{-2d_U(1+d_V)}\log(\varepsilon^{-1})\left[(1+d_V)d_U + d_V\right] \\
&\asymp \varepsilon^{-[\frac{(1+d_V)d_U}{2}+(1+d_V)]\varepsilon^{-(1+d_V)d_U}-2d_U(1+d_V)}\log(\varepsilon^{-1})\left[(1+d_V)d_U + d_V\right].
\end{aligned}$$

Taking logarithms on each side leads to:

$$\begin{aligned}
\log(N_\#) &\asymp \left(\left[\frac{(1+d_V)d_U}{2} + (1+d_V)\right]\varepsilon^{-(1+d_V)d_U} + 2d_U(1+d_V)\right)\log(\varepsilon^{-1}) + \log(\log(\varepsilon^{-1})) \\
&\asymp \left(\left[\frac{(1+d_V)d_U}{2} + (1+d_V)\right]\varepsilon^{-(1+d_V)d_U}\right)\log(\varepsilon^{-1}) \\
&=: \theta\varepsilon^{-\gamma}\log(\varepsilon^{-1}).
\end{aligned}$$

In fact, this is equivalent to

$$\frac{\gamma}{\theta}\log(N_\#) \asymp \log(\varepsilon^{-\gamma})\varepsilon^{-\gamma}$$

and, with the change of variable $t = \log(\varepsilon^{-1})$, we obtain

$$\frac{\gamma}{\theta}\log(N_\#) \asymp \gamma t e^{\gamma t}.$$

Using the Lambert W function [50] (defined by $W(z)\,e^{W(z)} = z$), we obtain that

$$t \asymp \frac{1}{\gamma}W\left(\frac{\gamma}{\theta}\log\left(N_\#\right)\right)$$

which leads to

$$\varepsilon \asymp \exp\left(-\frac{1}{\gamma}W\left(\frac{\gamma}{\theta}\log\left(N_\#\right)\right)\right).$$

For large arguments, $W(z) = \log(z) - \log(\log z) + o(1)$ (see [50, Section 4.1.4]), thus

$$W\left(\frac{\gamma}{\theta}\log\left(N_\#\right)\right) = \log(\log(N_\#)) - \log(\log(\log(N_\#))) + o(1),$$

and hence

$$\varepsilon \asymp \left(\frac{\log N_\#}{\log\log N_\#}\right)^{-1/\gamma} = \left(\frac{\log N_\#}{\log\log N_\#}\right)^{-\frac{1}{(1+d_V)d_U}}.$$

Remark 3.15 (Dependence of parameter complexity on approximation order). Similarly to Remark 3.14, we now express the approximation error of the network in Eq. (13) as a function of the total number of parameters, but using the alternative approximation order of Remark 3.13. We note that $n_{c_U} = \mathcal{O}(\varepsilon^{-d_U})$ by Lemma [41, Lemma 2].

In particular, we have

$$\begin{aligned}\log(N^{d_V}) &= d_V\left[(n_{c_U}+1)\log(2)+\log(C\sqrt{d_V})+n_{c_U}\log(C')+\frac{n_{c_U}}{2}\log(n_{c_U})+(1+n_{c_U})\log(\varepsilon^{-1})\right]\\ &\asymp d_V\left[\varepsilon^{-d_U}\frac{d_U}{2}\log(\varepsilon^{-1})+\varepsilon^{-d_U}\log(\varepsilon^{-1})\right]\\ &= \varepsilon^{-d_U}\log(\varepsilon^{-1})\left(\frac{d_U}{2}+1\right)d_V\end{aligned}$$

which implies

$$N^{d_V} \asymp \varepsilon^{-\varepsilon^{-d_U}d_V\left(\frac{d_U}{2}+1\right)}.$$

For K_1, we have:

$$\begin{aligned}K_1 &\asymp d_V^2(n_{c_U}+1)\log(\varepsilon^{-1})+d_V^2(n_{c_U}+1)\log(2)+d_V^2 n_{c_U}\log(C')+d_V^2\frac{n_{c_U}}{2}\log(n_{c_U})\\ &\asymp d_V^2\varepsilon^{-d_U}\log(\varepsilon^{-1})+d_V^2\varepsilon^{-d_U}\frac{d_U}{2}\log(\varepsilon^{-1})\\ &= \varepsilon^{-d_U}\log(\varepsilon^{-1})d_V^2\left(1+\frac{d_U}{2}\right)\end{aligned}$$

from which we deduce that

$$N^{d_V}K_1 \asymp \varepsilon^{-\varepsilon^{-d_U}d_V\left(\frac{d_U}{2}+1\right)-d_U}\log(\varepsilon^{-1})d_V^2\left(1+\frac{d_U}{2}\right).$$

Similarly, we have

$$\begin{aligned}\log(H^{n_{c_U}}) &\asymp n_{c_U}\left[\log(\sqrt{n_{c_U}})+\log(\varepsilon^{-1})\right]\\ &= \varepsilon^{-d_U}\left[\frac{d_U}{2}\log(\varepsilon^{-1})+\log(\varepsilon^{-1})\right]\\ &= \varepsilon^{-d_U}\log(\varepsilon^{-1})\left(1+\frac{d_U}{2}\right)\end{aligned}$$

hence,

$$H^{n_{c_U}} \asymp \varepsilon^{-\varepsilon^{-d_U}\left(1+\frac{d_U}{2}\right)}.$$

This implies

$$\begin{aligned}H^{n_{c_U}}K_2 &\asymp \varepsilon^{-\varepsilon^{-d_U}\left(1+\frac{d_U}{2}\right)}\left[\varepsilon^{-2d_u}d_U\log(\varepsilon^{-1})+\varepsilon^{-2d_U}\log(\varepsilon^{-1})\right]\\ &= \varepsilon^{-\varepsilon^{-d_U}\left(1+\frac{d_U}{2}\right)-2d_U}\log(\varepsilon^{-1})\left(1+d_U\right)\end{aligned}$$

and consequently:

$$N_{\#} \asymp \begin{cases}\varepsilon^{-\varepsilon^{-d_U}d_V\left(\frac{d_U}{2}+1\right)-d_U}\log(\varepsilon^{-1})d_V^2\left(1+\frac{d_U}{2}\right) & \text{if } d_V>1\\ \varepsilon^{-\varepsilon^{-d_U}\left(1+\frac{d_U}{2}\right)-2d_U}\log(\varepsilon^{-1})\left(1+d_U\right) & \text{if } d_V=1.\end{cases}$$

As our analysis in Remark 3.14 shows, the only parameter determining the leading order of the final rates is the power of the ε-power term, i.e. d_U. We conclude that

$$\varepsilon \asymp \left(\frac{\log N_{\#}}{\log\log N_{\#}}\right)^{-\frac{1}{d_U}}.$$

We note that reversing the approximation order yields a more favorable parameter scaling, as reflected by the improved rate, since $1/d_U > 1/(d_U(1+d_V))$. This observation underscores the importance of architectural design choices in determining the overall efficiency of operator approximation. We summarize these observations in Table 5.

	Theorem 3.8	Remark 3.13
Approximation goal	Establish scaling laws for a Lipschitz operator $G : U \mapsto V$	
Approximating architecture	$\sum_{\ell=1}^{N^{d_V}} \sum_{k=1}^{H^{n_{c_U}}} G^{(j)}[u_k](v_\ell) b_k(\mathbf{u}) \tau_\ell(x)$	
Approximation order	Function, then Functional	Functional, then Function
Value of n_{c_U}	$\mathcal{O}(\varepsilon^{-(1+d_V)d_U})$	$\mathcal{O}(\varepsilon^{-d_U})$
Value of N	$2C\sqrt{d_V}\varepsilon^{-1}$	$2^{n_{c_U}+1} C\sqrt{d_V}(C'\sqrt{n_{c_U}})^{n_{c_U}} \varepsilon^{-(1+n_{c_U})}$
Network class for τ_ℓ	$\mathcal{F}_{\mathrm{NN}}(d_V, 1, L, p, K, \kappa, R)$ with parameters scaling as $L = \mathcal{O}(d_V^2 \log d_V + d_V^2 \log(\varepsilon^{-1}))$, $p = \mathcal{O}(1)$, $K = \mathcal{O}(d_V^2 \log d_V + d_V^2 \log(\varepsilon^{-1}))$, $\kappa = \mathcal{O}(d_V^{d_V/2+1} \varepsilon^{-d_V-1})$, $R = 1$	$\mathcal{F}_{\mathrm{NN}}(d_V, 1, L, p, K, \kappa, R)$ with parameters scaling as $L = \mathcal{O}(d_V^2 \log d_V + d_V^2 \log(\varepsilon^{-(n_{c_U}+1)}) + d_V^2 \log(2^{n_{c_U}+1}(C'\sqrt{n_{c_U}})^{n_{c_U}}))$, $p = \mathcal{O}(1)$, $K = \mathcal{O}(d_V^2 \log d_V + d_V^2 \log(\varepsilon^{-(n_{c_U}+1)}) + d_V^2 \log(2^{n_{c_U}+1}(C'\sqrt{n_{c_U}})^{n_{c_U}}))$, $\kappa = \mathcal{O}(d_V^{d_V/2+1} \varepsilon^{-(d_V+1)(n_{c_U}+1)} \times [2^{n_{c_U}+1}(C'\sqrt{n_{c_U}})^{n_{c_U}}]^{-1})$, $R = 1$
Value of H	$2^{d_V+1} C'\sqrt{n_{c_U}}(C\sqrt{d_V})^{d_V} \varepsilon^{-(d_V+1)}$	$2C'\sqrt{n_{c_U}}\varepsilon^{-1}$
Network class for b_k	$\mathcal{F}_{\mathrm{NN}}(n_{c_U}, 1, L, p, K, \kappa, R)$ with parameters scaling as $L = \mathcal{O}(n_{c_U}^2 \log n_{c_U} + n_{c_U}^2 (d_V+1)\log(\varepsilon^{-1}) + n_{c_U}^2 \log(2^{d_V+1}(C\sqrt{d_V})^{d_V}))$, $p = \mathcal{O}(1)$, $K = \mathcal{O}(n_{c_U}^2 \log n_{c_U} + n_{c_U}^2 (d_V+1)\log(\varepsilon^{-1}) + n_{c_U}^2 \log(2^{d_V+1}(C\sqrt{d_V})^{d_V}))$, $\kappa = \mathcal{O}(n_{c_U}^{n_{c_U}/2+1} \varepsilon^{-(d_V+1)(n_{c_U}+1)} \times [2^{d_V+1}(C\sqrt{d_V})^{d_V}]^{n_{c_U}+1})$, $R = 1$	$\mathcal{F}_{\mathrm{NN}}(n_{c_U}, 1, L, p, K, \kappa, R)$ with parameters scaling as $L = \mathcal{O}(n_{c_U}^2 \log(n_{c_U}) + n_{c_U}^2 \log(\varepsilon^{-1}))$, $p = \mathcal{O}(1)$, $K = \mathcal{O}(n_{c_U}^2 \log(n_{c_U}) + n_{c_U}^2 \log(\varepsilon^{-1}))$, $\kappa = \mathcal{O}(n_{c_U}^{n_{c_U}/2+1} \varepsilon^{-(n_{c_U}+1)})$, $R = 1$
Total parameters $N_\#$ scaling	$\left(\frac{\log N_\#}{\log\log N_\#}\right)^{-\frac{1}{(1+d_V)d_U}}$	$\left(\frac{\log N_\#}{\log\log N_\#}\right)^{-\frac{1}{d_U}}$

Table 5: Comparison of scaling behaviors for the space-approximation networks τ_ℓ, function-approximation networks b_k and total number of parameters under different approximation orders for operator learning. We have $n_{c_U} = \mathcal{O}(\delta^{-d_U})$ by [41, Lemma 2]. The results illustrate that (1) scaling complexity can be redistributed between the subnetworks without affecting the expressive power of the overall architecture, and (2) the chosen approximation order directly impacts the scaling of the total number of parameters.

Combining previous results, we conclude with the scaling laws for the multiple operator approximation problem. In particular, the proof reduces multiple operator learning to learning a finite amount of single operators.

Theorem 3.16 (Multiple Operator Scaling Laws)**.** *Let $d_W, d_U, d_V > 0$ be integers,*

$$\gamma_W, \gamma_U, \gamma_V, \beta_W, \beta_U, \beta_V, L_W, L_U, L_V, L_G, L_{\mathcal{G}} > 0 \qquad \text{and} \qquad r_G, r_{\mathcal{G}} \geq 1$$

*and assume that $W(d_W, \gamma_W, L_W, \beta_W)$, $U(d_U, \gamma_U, L_U, \beta_U)$ and $V(d_V, \gamma_V, L_V, \beta_V)$ satisfy Assumption **S.4**. Let G be a map such that*

$$\begin{aligned} &G : \{\alpha : \Omega_W \mapsto \mathbb{R} \,|\, \|\alpha\|_{\mathrm{L}^\infty} \leq \beta_W\} \mapsto \mathcal{G} \qquad \text{where} \\ &\mathcal{G} = \Big\{ G[\alpha] \,|\, G[\alpha] : \{u : \Omega_U \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \leq \beta_U\} \mapsto V \text{ and} \\ &\qquad \|G[\alpha][u_1] - G[\alpha][u_2]\|_{\mathrm{L}^\infty(\Omega_V)} \leq L_{\mathcal{G}} \|u_1 - u_2\|_{\mathrm{L}^{r_{\mathcal{G}}}(\Omega_U)} \Big\} \end{aligned}$$

Furthermore, assume that G satisfies

$$\|G(\alpha_1) - G(\alpha_2)\|_{\mathrm{L}^\infty(\{u:\Omega_U \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \leq \beta_U\} \times \Omega_V)} \leq L_G \|\alpha_1 - \alpha_2\|_{\mathrm{L}^{r_G}(\Omega_U)} \tag{21}$$

for $\alpha_1, \alpha_2 \in \{\alpha : \Omega_W \mapsto \mathbb{R} \,|\, \|\alpha\|_{\mathrm{L}^\infty} \leq \beta_W\}$.

There exists constants C depending on γ_V, L_V, C_δ depending on $L_{\mathcal{G}}, d_U, \gamma_U, r_{\mathcal{G}}, L_U$, C' depending on $\beta_U, L_{\mathcal{G}}, d_U, \gamma_U, r_{\mathcal{G}}$, C_ς depending on $L_G, d_W, \gamma_W, r_G, L_W$ and C'' depending on $\beta_W, L_G, d_W, \gamma_W, r_G$ such that the following holds. For any $\varepsilon > 0$,

- *let $N = 2^{n_{c_W}+2} C \sqrt{d_V} (C'' \sqrt{n_{c_W}})^{n_{c_W}} \varepsilon^{-(n_{c_W}+1)}$ and consider the network class $\mathcal{F}_1 = \mathcal{F}_{\mathrm{NN}}(d_V, 1, L_1, p_1, K_1, \kappa_1, R_1)$ with parameters scaling as*

$$\begin{aligned} L_1 &= \mathcal{O}\left(d_V^2 \log d_V + d_V^2 (n_{c_W}+1) \log(\varepsilon^{-1}) + d_V^2 \log(2^{n_{c_W}+1} (C'' \sqrt{n_{c_W}})^{n_{c_W}})\right), \quad p_1 = \mathcal{O}(1), \\ K_1 &= \mathcal{O}\left(d_V^2 \log d_V + d_V^2 (n_{c_W}+1) \log(\varepsilon^{-1}) + d_V^2 \log(2^{n_{c_W}+1} (C'' \sqrt{n_{c_W}})^{n_{c_W}})\right) \\ \kappa_1 &= \mathcal{O}(d_V^{d_V/2+1} \varepsilon^{-(d_V+1)(n_{c_W}+1)} \left[2^{n_{c_W}+1} (C'' \sqrt{n_{c_W}})^{n_{c_W}}\right]^{(d_V+1)}), \qquad R_1 = 1 \end{aligned}$$

  where the constants hidden in $\mathcal{O}$ depend on γ_V and L_V;

- *let $\{v_\ell\}_{\ell=1}^{N^{d_V}} \subset \Omega_V$ be a uniform grid with spacing $2\gamma_V / N$ along each dimension;*

- *let $\delta = \frac{C_\delta \varepsilon^{(1+d_V)(1+n_{c_W})}}{2^{d_V+n_{c_W}+2} (C\sqrt{d_V})^{d_V} (C'' \sqrt{n_{c_W}})^{n_{c_W}}}$ and let $\{c_m\}_{m=1}^{n_{c_U}} \subset \Omega_U$ be points so that $\{\mathcal{B}_\delta(c_m)\}_{m=1}^{n_{c_U}}$ is a cover of Ω_U for some n_{c_U};*

- *let $H = 2^{(d_V+1)(n_{c_W}+2)} C' \sqrt{n_{c_U}} (C\sqrt{d_V})^{d_V} (C'' \sqrt{n_{c_W}})^{n_{c_W}(d_V+1)} \varepsilon^{-(d_V+1)(1+n_{c_W})}$ and consider the network class $\mathcal{F}_2 = \mathcal{F}_{\mathrm{NN}}(n_{c_U}, 1, L_2, p_2, K_2, \kappa_2, R_2)$ with parameters scaling as*

$$\begin{aligned} L_2 &= \mathcal{O}\big(n_{c_U}^2 \log n_{c_U} + n_{c_U}^2 (d_V+1)(n_{c_W}+1) \log(\varepsilon^{-1}) + n_{c_U}^2 \log(2^{d_V+1} (C\sqrt{d_V})^{d_V}) \\ &\qquad + n_{c_U}^2 (d_V+1) \log(2^{n_{c_W}+1} (C'' \sqrt{n_{c_W}})^{n_{c_W}})\big), \quad p_2 = \mathcal{O}(1), \\ K_2 &= \mathcal{O}\big(n_{c_U}^2 \log n_{c_U} + n_{c_U}^2 (d_V+1)(n_{c_W}+1) \log(\varepsilon^{-1}) + n_{c_U}^2 \log(2^{d_V+1} (C\sqrt{d_V})^{d_V}) \\ &\qquad + n_{c_U}^2 (d_V+1) \log(2^{n_{c_W}+1} (C'' \sqrt{n_{c_W}})^{n_{c_W}})\big), \\ \kappa_2 &= \mathcal{O}(n_{c_U}^{n_{c_U}/2+1} \varepsilon^{-(d_V+1)(n_{c_U}+1)(n_{c_W}+1)} [2^{d_V+1} (C\sqrt{d_V})^{d_V}]^{n_{c_U}+1} \left[2^{d_V+1} (C\sqrt{d_V})^{d_V}\right]^{(d_V+1)(n_{c_U}+1)}), \\ R_2 &= 1 \end{aligned}$$

  where the constants hidden in $\mathcal{O}$ depend on $\beta_U, L_{\mathcal{G}}, d_U, \gamma_U, r_{\mathcal{G}}$;

- *let $\zeta = C_\varsigma \varepsilon$ and let $\{y_m\}_{m=1}^{n_{c_W}} \subset \Omega_W$ be points so that $\{\mathcal{B}_\zeta(y_m)\}_{m=1}^{n_{c_W}}$ is a cover of Ω_W for some n_{c_W};*

- *let* $P = 2C''\sqrt{n_{c_W}}\varepsilon^{-1}$ *and consider the network class* $\mathcal{F}_3 = \mathcal{F}_{\mathrm{NN}}(n_{c_W}, 1, L_3, p_3, K_3, \kappa_3, R_3)$ *with parameters scaling as*

$$L_3 = \mathcal{O}\left(n_{c_W}^2 \log(n_{c_W}) + n_{c_W}^2 \log(\varepsilon^{-1})\right), \quad p_3 = \mathcal{O}(1), \quad K_3 = \mathcal{O}\left(n_{c_W}^2 \log n_{c_W} + n_{c_W}^2 \log(\varepsilon^{-1})\right),$$
$$\kappa_3 = \mathcal{O}(n_{c_W}^{n_{c_W}/2+1}\varepsilon^{-n_{c_W}-1}), \qquad R_3 = 1$$

where the constants hidden in $\mathcal{O}$ *depend on* $\beta_W, L_G, d_W, \gamma_W, r_G$.

Then, there exists networks $\{\tau_\ell\}_{\ell=1}^{N^{d_V}} \subset \mathcal{F}_1$, *networks* $\{b_k\}_{k=1}^{H^{n_{c_U}}} \subset \mathcal{F}_2$, *networks* $\{l_p\}_{p=1}^{P} \subset \mathcal{F}_3$, *functions* $\{u_k\}_{k=1}^{H^{n_{c_U}}} \subset \{u : \Omega_U \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \leq \beta_U\}$ *and functions* $\{\alpha_p\}_{p=1}^{P} \subset \{\alpha : \Omega_W \mapsto \mathbb{R} \,|\, \|\alpha\|_{\mathrm{L}^\infty} \leq \beta_W\}$ *such that*

$$\sup_{\alpha\in W}\sup_{u\in U}\sup_{x\in\Omega_V}\left|G[\alpha][u](x) - \sum_{p=1}^{P^{n_{c_W}}}\sum_{k=1}^{H^{n_{c_U}}}\sum_{\ell=1}^{N^{d_V}} G[\alpha_p][u_k](v_\ell) l_p(\boldsymbol{\alpha}) b_k(\mathbf{u})\tau_\ell(x)\right| \leq \varepsilon, \tag{22}$$

where $\boldsymbol{\alpha} = (\alpha(y_1), \alpha(y_2), ..., \alpha(y_{n_{c_W}}))^\top$ *is a discretization of* α *and* $\mathbf{u} = (u(c_1), u(c_2), ..., u(c_{n_{c_U}}))^\top$ *is a discretization of* u.

The proof of the theorem is presented in Section 4.3.2. As in the operator learning setting of Theorem 3.8, the argument follows an inherently sequential structure, proceeding through successive approximation stages: first for the mapping $\alpha \mapsto G[\alpha]$, then for $u \mapsto G[\alpha][u]$, and finally for $x \mapsto G[\alpha][u](x)$. As a result, the scaling behavior deteriorates progressively, since each stage of the approximation inherits and compounds the error and complexity of the preceding one. This increasing scaling complexity is reflected in the growth of the network classes $\mathcal{F}_3$, $\mathcal{F}_1$, and $\mathcal{F}_2$.

Remark 3.17 (Total number of parameter for multiple operator learning). Similarly to Remark 3.14, we now express the approximation error of the network in Eq. (22) as a function of the total number of parameters $N_\# = P^{n_{c_W}} K_3 + N^{d_V} K_1 + H^{n_{c_U}} K_2$. We note that $n_{c_W} = \mathcal{O}(\varepsilon^{-d_W})$ and $n_{c_U} = \mathcal{O}(\delta^{-d_U})$ by [41, Lemma 2]. For the latter, we compute

$$\begin{aligned}
\log(\delta^{-d_U}) &= -d_U \log\left(\frac{C_\delta \varepsilon^{(1+d_V)(1+n_{c_W})}}{2^{d_V+n_{c_W}+2}(C\sqrt{d_V})^{d_V}(C''\sqrt{n_{c_W}})^{n_{c_W}}}\right) \\
&= -d_u \log(C_\delta) + d_U(1+d_V)(1+n_{c_W})\log(\varepsilon^{-1}) + d_U(d_V+n_{c_W}+2)\log(2) \\
&+ d_U n_{c_W}\log(C'') + d_U\frac{n_{c_W}}{2}\log(n_{c_W}) \\
&\asymp d_U(1+d_V)(1+\varepsilon^{-d_W})\log(\varepsilon^{-1}) + d_U\varepsilon^{-d_W} + \frac{d_U d_W}{2}\varepsilon^{-d_W}\log(\varepsilon^{-1}) \\
&\asymp \log(\varepsilon^{-1})\varepsilon^{-d_W} d_U\left((1+d_V) + \frac{d_W}{2}\right)
\end{aligned}$$

which leads to

$$n_{c_U} \asymp \varepsilon^{-\varepsilon^{-d_W} d_U\left((1+d_V)+\frac{d_W}{2}\right)}.$$

We now consider:

$$\begin{aligned}
P^{n_{c_W}} K_3 &\asymp n_{c_W}^{n_{c_W}/2}\varepsilon^{-n_{c_W}}\left(n_{c_W}^2\log n_{c_W} + n_{c_W}^2\log(\varepsilon^{-1})\right) \\
&= \varepsilon^{-\varepsilon^{-d_W}\left(1+\frac{d_W}{2}\right)}\left(d_W\varepsilon^{-2d_W}\log(\varepsilon^{-1}) + \varepsilon^{-2d_W}\log(\varepsilon^{-1})\right) \\
&\asymp \varepsilon^{-\varepsilon^{-d_W}\left(1+\frac{d_W}{2}\right)-2d_W}\log(\varepsilon^{-1})(1+d_W).
\end{aligned} \tag{23}$$

Next, we note that

$$\begin{aligned}
\log(N^{d_V}) &= d_V\log(2^{n_{c_W}+2}C\sqrt{d_V}(C''\sqrt{n_{c_W}})^{n_{c_W}}\varepsilon^{-(n_{c_W}+1)}) \\
&= d_V\left[(n_{c_W}+1)\log(2) + \log(C\sqrt{d_V}) + n_{c_W}\log(C'') + \frac{n_{c_W}}{2}\log(n_{c_W}) + (n_{c_W}+1)\log(\varepsilon^{-1})\right]
\end{aligned}$$

$$
\begin{aligned}
&\asymp d_V \left[\varepsilon^{-d_V}(\log(2)+C'') + \varepsilon^{-d_W}\log(\varepsilon^{-1})\left(\frac{d_W}{2}+1\right) + \log(\varepsilon^{-1})\right] \\
&= d_V \varepsilon^{-d_W}\log(\varepsilon^{-1})\left(\frac{d_W}{2}+1\right)
\end{aligned}
$$

which implies

$$
N^{d_V} \asymp \varepsilon^{-\varepsilon^{-d_W} d_V\left(\frac{d_W}{2}+1\right)}.
$$

Using this, we have

$$
\begin{aligned}
N^{d_V}K_1 &\asymp \varepsilon^{-\varepsilon^{-d_W} d_V\left(\frac{d_W}{2}+1\right)}\left[d_V^2\log d_V + d_V^2(n_{c_W}+1)\log(\varepsilon^{-1}) + d_V^2\log(2^{n_{c_W}+1}(C''\sqrt{n_{c_W}})^{n_{c_W}})\right] \\
&\asymp \varepsilon^{-\varepsilon^{-d_W} d_V\left(\frac{d_W}{2}+1\right)}\Bigg[d_V^2\varepsilon^{-d_W}\log(\varepsilon^{-1}) + d_V^2\log(\varepsilon^{-1}) + d_V^2(n_{c_W}+1)\log(2) \\
&\quad + d_V^2 n_{c_W}\log(C'') + d_V^2\frac{n_{c_W}}{2}\log(n_{c_W}))\Bigg] \\
&\asymp \varepsilon^{-\varepsilon^{-d_W} d_V\left(\frac{d_W}{2}+1\right)}\left[d_V^2\varepsilon^{-d_W}\log(\varepsilon^{-1}) + d_V^2\frac{d_W}{2}\varepsilon^{-d_W}\log(\varepsilon^{-1})\right] \\
&= \varepsilon^{-\varepsilon^{-d_W} d_V\left(\frac{d_W}{2}+1\right)-d_W}\log(\varepsilon^{-1})d_V^2\left(1+\frac{d_W}{2}\right). \qquad (24)
\end{aligned}
$$

Then, we consider

$$
\begin{aligned}
&\log(H^{n_{c_U}}) = n_{c_U}\log(2^{(d_V+1)(n_{c_W}+2)}C'\sqrt{n_{c_U}}(C\sqrt{d_V})^{d_V}(C''\sqrt{n_{c_W}})^{n_{c_W}(d_V+1)}\varepsilon^{-(d_V+1)(1+n_{c_W})}) \\
&= n_{c_U}\Bigg[(d_V+1)(n_{c_W}+2)\log(2) + \log(C'(C\sqrt{d_V})^{d_V}) + \frac{1}{2}\log(n_{c_u}) + n_{c_W}(d_V+1)\log(C'') \\
&+ n_{c_W}\frac{(d_V+1)}{2}\log(n_{c_W}) + (d_V+1)(1+n_{c_W})\log(\varepsilon^{-1})\Bigg] \\
&\asymp n_{c_U}\Bigg[(d_V+1)\varepsilon^{-d_W}\log(2) + d_U\left((1+d_V)+\frac{d_W}{2}\right)\varepsilon^{-d_W}\log(\varepsilon^{-1}) + \varepsilon^{-d_W}(d_V+1)\log(C'') \\
&+ d_W\frac{(d_V+1)}{2}\varepsilon^{-d_W}\log(\varepsilon^{-1}) + (d_V+1)\varepsilon^{-d_W}\log(\varepsilon^{-1})\Bigg] \\
&\asymp n_{c_U}\left[\varepsilon^{-d_W}\log(\varepsilon^{-1})\left(d_U\left((1+d_V)+\frac{d_W}{2}\right) + d_W\frac{(d_V+1)}{2} + (d_V+1)\right)\right] \\
&\asymp \varepsilon^{-\varepsilon^{-d_W} d_U\left((1+d_V)+\frac{d_W}{2}\right)-d_W}\log(\varepsilon^{-1})\left[d_U\left((1+d_V)+\frac{d_W}{2}\right) + d_W\frac{(d_V+1)}{2} + (d_V+1)\right]
\end{aligned}
$$

which implies that

$$
H^{n_{c_U}} \asymp \varepsilon^{-\varepsilon^{-d_W d_U\left((1+d_V)+\frac{d_W}{2}\right)-d_W}\left[\left(d_U\left((1+d_V)+\frac{d_W}{2}\right)+d_W\frac{(d_V+1)}{2}+(d_V+1)\right)\right]}.
$$

We also note that

$$
\begin{aligned}
K_2 &\asymp n_{c_U}^2\log n_{c_U} + n_{c_U}^2(d_V+1)(n_{c_W}+1)\log(\varepsilon^{-1}) + n_{c_U}^2\log(2^{d_V+1}(C\sqrt{d_V})^{d_V}) \\
&\quad + n_{c_U}^2(d_V+1)\log(2^{n_{c_W}+1}(C''\sqrt{n_{c_W}})^{n_{c_W}}) \\
&\asymp n_{c_U}^2\log n_{c_U} + n_{c_U}^2(d_V+1)n_{c_W}\log(\varepsilon^{-1}) + n_{c_U}^2(d_V+1)(n_{c_W}+1)\log(2) \\
&\quad + n_{c_U}^2(d_V+1)n_{c_W}\log(C'') + n_{c_U}^2(d_V+1)\frac{n_{c_W}}{2}\log(n_{c_W}) \\
&\asymp n_{c_U}^2\log n_{c_U} + n_{c_U}^2(d_V+1)\varepsilon^{-d_W}\log(\varepsilon^{-1}) + n_{c_U}^2(d_V+1)\frac{d_W}{2}\varepsilon^{-d_W}\log(\varepsilon^{-1}) \\
&= n_{c_U}^2\log n_{c_U} + n_{c_U}^2\varepsilon^{-d_W}\log(\varepsilon^{-1})(d_V+1)\left(1+\frac{d_W}{2}\right)
\end{aligned}
$$

$$
\begin{aligned}
&\asymp \varepsilon^{-2\varepsilon^{-d_W} d_U \left((1+d_V)+\frac{d_W}{2}\right)-d_W} \log(\varepsilon^{-1}) d_U \left((1+d_V)+\frac{d_W}{2}\right) \\
&+ \varepsilon^{-2\varepsilon^{-d_W} d_U \left((1+d_V)+\frac{d_W}{2}\right)-d_W} \log(\varepsilon^{-1})(d_V+1)\left(1+\frac{d_W}{2}\right) \\
&= \varepsilon^{-2\varepsilon^{-d_W} d_U \left((1+d_V)+\frac{d_W}{2}\right)-d_W} \log(\varepsilon^{-1}) \left[d_U \left((1+d_V)+\frac{d_W}{2}\right)+(d_V+1)\left(1+\frac{d_W}{2}\right)\right]
\end{aligned}
$$

which therefore yields:

$$
\begin{aligned}
H^{n_{c_U}} K_2 &\asymp \varepsilon^{-\varepsilon^{-\varepsilon^{-d_W} d_U \left((1+d_V)+\frac{d_W}{2}\right)-d_W} \left[\left(d_U \left((1+d_V)+\frac{d_W}{2}\right)+d_W \frac{(d_V+1)}{2}+(d_V+1)\right)\right]-2\varepsilon^{-d_W} d_U \left((1+d_V)+\frac{d_W}{2}\right)-d_W} \\
&\times \log(\varepsilon^{-1}) \left[d_U \left((1+d_V)+\frac{d_W}{2}\right)+(d_V+1)\left(1+\frac{d_W}{2}\right)\right]. \qquad (25)
\end{aligned}
$$

Combining Eqs. (23), (24) and (25), we conclude that

$$
\begin{aligned}
N_\# &\asymp \varepsilon^{-\varepsilon^{-\varepsilon^{-d_W} d_U \left((1+d_V)+\frac{d_W}{2}\right)-d_W} \left[\left(d_U \left((1+d_V)+\frac{d_W}{2}\right)+d_W \frac{(d_V+1)}{2}+(d_V+1)\right)\right]-2\varepsilon^{-d_W} d_U \left((1+d_V)+\frac{d_W}{2}\right)-d_W} \\
&\quad \times \log(\varepsilon^{-1}) \left[d_U \left((1+d_V)+\frac{d_W}{2}\right)+(d_V+1)\left(1+\frac{d_W}{2}\right)\right] \\
&\quad =: \varepsilon^{-\gamma_2 \varepsilon^{-\gamma_1 \varepsilon^{-d_W}-d_W}-\gamma_3 \varepsilon^{-d_W}-d_W} \log(\varepsilon^{-1})\gamma_4.
\end{aligned}
$$

We therefore have

$$
\log(N_\#) \asymp \left(\gamma_2 \varepsilon^{-\gamma_1 \varepsilon^{-d_W}-d_W}+\gamma_3 \varepsilon^{-d_W}-d_W\right)+\log(\log(\varepsilon^{-1}))+\log(\gamma_4) \asymp \gamma_2 \varepsilon^{-\gamma_1 \varepsilon^{-d_W}-d_W}.
$$

Taking an additional logarithm, we obtain

$$
\log(\log(N_\#)) \asymp \log(\gamma_2)+\left(\gamma_1 \varepsilon^{-d_W}-d_W\right)\log(\varepsilon^{-d}) \asymp \gamma_1 \varepsilon^{-d_W} \log(\varepsilon^{-d}).
$$

Proceeding as in Remark 3.14, with the Lambert function inversion, this yields the final scaling

$$
\varepsilon \asymp \left(\frac{\log\log N_\#}{\log\log\log N_\#}\right)^{-1/d_W}.
$$

As expected, moving from the single operator to the multiple operator setting incurs a less favorable scaling of the total number of parameters (see Table 5), consistent with the higher representational complexity required.

Remark 3.18 (Improved rates with additional low-dimensional structure). If the input function spaces W and U admit a finite orthonormal basis representations and the discretization grids satisfy stable linear reconstruction properties as in [41, Assumption 4], one can expect substantially improved approximation rates. In particular, under these additional structural assumptions, one should observe at least a transition from double–iterated to single–iterated logarithmic convergence, potentially recovering the rates observed for single–operator learning in the general setting (see Remarks 3.14 and 3.15, and Table 5).

Remark 3.19 (A review of different multiple operator network architectures). By combining the proof of Theorem 3.16, Remark 3.9 and Remark 3.11, we can prove scaling laws for various network architectures depending on the assumptions we make on the map G. In particular, we replicate the proof of Theorem 3.16 for G satisfying

$$
\begin{aligned}
&G : \{\alpha : \Omega_W \mapsto \mathbb{R} \,|\, \|\alpha\|_{\mathrm{L}^\infty} \leq \beta_W\} \mapsto \mathcal{G} \qquad \text{where} \\
&\mathcal{G} = \Big\{G[\alpha] \,|\, G[\alpha] : \{u : \Omega_U^{(\alpha)} \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \leq \beta_{U^{(\alpha)}}\} \mapsto V^{(\alpha)} \text{ and} \\
&\quad \|G[\alpha][u_1]-G[\alpha][u_2]\|_{\mathrm{L}^\infty(\Omega_{V^{(\alpha)}})} \leq L^{(\alpha)} \|u_1-u_2\|_{\mathrm{L}^{r^\alpha}(\Omega_{U^\alpha})}\Big\}.
\end{aligned}
$$

We start by considering the set $\mathcal{S} := \bigcup_{\alpha\in W}(U^\alpha \times \Omega_{V^{(\alpha)}})$ where $(U^\alpha \times V^{(\alpha)})$ are such that $G[\alpha] : \{u : \Omega_U^{(\alpha)} \mapsto \mathbb{R} \,|\, \|u\|_{\mathrm{L}^\infty} \leq \beta_{U^{(\alpha)}}\} \mapsto V^{(\alpha)}$ for $\alpha \in W$. Furthermore, assume that G satisfies

$$\|G(\alpha_1) - G(\alpha_2)\|_{\mathrm{L}^\infty(\mathcal{S})} \leq L_G \|\alpha_1 - \alpha_2\|_{\mathrm{L}^{r_G}(\Omega_U)}$$

for $\alpha_1, \alpha_2 \in \{\alpha : \Omega_W \mapsto \mathbb{R} \,|\, \|\alpha\|_{\mathrm{L}^\infty} \leq \beta_W\}$. By the axiom of choice, we can select an element $s \in \mathcal{S}$ such that $s^{(\alpha)} = (u^{(\alpha)}, x^{(\alpha)}) \in (U^\alpha \times \Omega_{V^{(\alpha)}})$. Then, we define the functional $f(\alpha) = G[\alpha][u^{(\alpha)}](x^\alpha)$ and, by the above assumption on Lipschitz continuity of G, we deduce that $f : \{\alpha : \Omega_W \mapsto \mathbb{R} \,|\, \|\alpha\|_{\mathrm{L}^\infty} \leq \beta_W\} \mapsto \mathbb{R}$ is Lipschitz. We apply Theorem 3.6 to obtain that

$$\sup_{\alpha\in W} \left| f(\alpha) - \sum_{p=1}^{P^{n_{c_W}}} f(\alpha_k) l_p(P_{\mathcal{C}_W}(\alpha)) \right| = \sup_{\alpha\in U} \left| G[\alpha][u^{(\alpha)}](x^{(\alpha)}) - \sum_{p=1}^{P^{n_{c_W}}} G[\alpha_p][u^{(\alpha_p)}](x^{(\alpha_p)}) l_p(P_{\mathcal{C}_W}(\alpha)) \right| \leq \frac{\varepsilon}{2}.$$

It now remains to approximate the p operators $G[\alpha_p]$ by any of the architectures in Remark 3.9 or Remark 3.11. We summarize the final multiple operator architectures in Tables 6 and 7 (only for the first alternative formulation). From the latter, we note that our scaling laws can be transferred, in particular, to the MNO and MIONet [25] architectures.

Network type \ Assumptions	$U^{(\alpha_p)}$ and $V^{(\alpha_p)}$ distinct	$U^{(\alpha_p)} = U$
Exact	$\displaystyle\sum_{p=1}^{P^{n_{c_W}}} \sum_{\ell=1}^{(N^{(p)})^{d_{V^{(p)}}}} \sum_{k=1}^{H^{(p)}} G[\alpha_p][u_k^{(p)}](v_\ell^{(p)})\, l_p(\boldsymbol{\alpha})\, b_{pk}(\mathbf{u}^{(p)})\, \tau_{p\ell}(x)$	$\displaystyle\sum_{p=1}^{P^{n_{c_W}}} \sum_{\ell=1}^{(N^{(p)})^{d_{V^{(p)}}}} \sum_{k=1}^{H^{n_{c_U}}} G[\alpha_p][u_k](v_\ell^{(p)})\, l_p(\boldsymbol{\alpha})\, b_k(\mathbf{u})\, \tau_{p\ell}(x)$
Alternative	$\displaystyle\sum_{p=1}^{P^{n_{c_W}}} \sum_{k=1}^{H^{(p)}} l_p(\boldsymbol{\alpha})\, b_{pk}(\mathbf{u}^{(p)})\, \hat{\tau}_{pk}(x)$	$\displaystyle\sum_{p=1}^{P^{n_{c_W}}} \sum_{k=1}^{H^{n_{c_U}}} l_p(\boldsymbol{\alpha})\, b_k(\mathbf{u})\, \hat{\tau}_{pk}(x)$

Table 6: Multiple operator network architectures with distinct or partially fixed $U^{(\alpha_p)}$, $V^{(\alpha_p)}$. We write $H^{(p)} = H^{n_{c_{U^{(\alpha_p)}}}}$ and $N^{(p)} = N^{(\alpha_p)}$.

Network type \ Assumptions	$V^{(\alpha_p)} = V$	$U^{(\alpha_p)} = U$ and $V^{(\alpha_p)} = V$
Exact	$\displaystyle\sum_{p=1}^{P^{n_{c_W}}} \sum_{k=1}^{H^{(p)}} \sum_{\ell=1}^{N^{d_V}} G[\alpha_p][u_k^{(p)}](v_\ell)\, l_p(\boldsymbol{\alpha})\, b_{pk}(\mathbf{u}^{(p)})\, \tau_\ell(x)$	$\displaystyle\sum_{p=1}^{P^{n_{c_W}}} \sum_{k=1}^{H^{n_{c_U}}} \sum_{\ell=1}^{N^{d_V}} G[\alpha_p][u_k](v_\ell)\, l_p(\boldsymbol{\alpha})\, b_k(\mathbf{u})\, \tau_\ell(x)$
Alternative	$\displaystyle\sum_{p=1}^{P^{n_{c_W}}} \sum_{k=1}^{H^{(p)}} l_p(\boldsymbol{\alpha})\, b_{pk}(\mathbf{u}^{(p)})\, \hat{\tau}_{pk}(x)$	$\displaystyle\sum_{p=1}^{P^{n_{c_W}}} \sum_{k=1}^{H^{n_{c_U}}} l_p(\boldsymbol{\alpha})\, b_k(\mathbf{u})\, \hat{\tau}_{pk}(x)$

Table 7: Multiple operator network architectures with partially or fully fixed $U^{(\alpha_p)}$, $V^{(\alpha_p)}$. We write $H^{(p)} = H^{n_{c_{U^{(\alpha_p)}}}}$ and $N^{(p)} = N^{(\alpha_p)}$.

4 Proofs

In this section, we present the proofs. We first establish two versions of the universal approximation theorem for multiple nonlinear operators, then address scaling laws for functional and single operator approximation. Finally, we conclude with the proof of the scaling laws in the multiple operator approximation setting.

4.1 Proof of Theorem 3.4

In this section, we prove the universal approximation property for the MNO and MONet networks in L^∞. The intuition behind the proof of Theorem 3.4 parallels our earlier discussion in Section 2.1. The key step is to sequentially separate the input variables of the operator G, thereby transforming the operator approximation problem into a sequence of function and functional approximation problems.

Proof of Theorem 3.4. We first observe that any multiple operator network of the form

$$\sum_{k=1}^{N}\sum_{\ell=1}^{M}\tau_k(x)\,b_{k\ell}(u)\,L_{k\ell}(\alpha)$$

can be re-indexed into a MNO of the form

$$\sum_{p=1}^{P}\sum_{k=1}^{H^{(p)}}l_p(\alpha)\,b_{pk}(u)\,\tau_{pk}(x),$$

where l_p, b_{pk}, and τ_{pk} retain the same structure as in Definition 3.1. Since the universal approximation statement in (7) does not rely on explicit scalings for N and M, this re-indexing leaves the result unaffected. Consequently, Theorem 3.4 may be established for one representation and inferred for the other. Without loss of generality, we therefore present the proof using the MONet architecture.

Let $\varepsilon > 0$. For an arbitrary, fixed $\alpha \in W$ and $u \in V$, consider the function $f(x) = G[\alpha][u](x)$. By assumption, $f \in \mathcal{F} := G[W][U]$ which is a compact subset of $V = \mathrm{C}^0(\Omega_V)$ and thus by Theorem 2.4, we can find $N \in \mathbb{N}, \eta_k \in \mathbb{R}, \omega_k \in \mathbb{R}^n$ such that:

$$\left| f(x) - \sum_{k=1}^{N} c_k(f(\cdot))\sigma(\omega_k \cdot x + \zeta_k) \right| < \varepsilon/3,$$

where $c_i(f)$ are continuous linear functions. The approximation results hold for all $f \in \mathcal{F}$, and since $f(\cdot) = G[\alpha][u](\cdot)$ the coefficients $c_k(f(\cdot)) = c_k(G[\alpha][u](\cdot))$ are continuous functionals mapping $W \times U \to \mathbb{R}$.

For each k and arbitrary fixed α, define $F^{(k)} : U \to \mathbb{R}$ by

$$F^{(k)}(u) := c_k(G[\alpha][u])$$

which is a continuous functional with respect to u. Similarly to [12], by the Tietze Extension theorem the functionals $F^{(k)}$ are extended to continuous functionals $F_*^{(k)}$ on all of U^* from Lemma 2.5, so that $F_*^{(k)}(u) = F^{(k)}(u)$ for all $u \in U$. Since U^* is a compact set, there exists $\delta > 0$ such that

$$\left| F_*^{(k)}(u_1) - F_*^{(k)}(u_2) \right| < \frac{\varepsilon}{6L_1}$$

for all $u_1, u_2 \in U^*$ with $\|u_1 - u_2\|_{\mathrm{C}^0(\Omega_U)} < \delta$, and $L_1 = \sum_{k=1}^{N} \sup_{x\in\Omega_V} |\sigma(\omega_k \cdot x + \zeta_k)|$. The extension is needed since the construction of the functions on the η_k-net by Equation 4 may reside in $U^* \setminus U$.

Let $\delta_k < \delta$, where δ_k is defined in (2). (Abusing notation, we select δ_k satisfying $\delta_k < \delta$, independently of the k^{th} element in the sequence of (2), while retaining the subscript for simplicity.) Then by Lemma 2.5(2), there exists $u_{\eta_k} \in U_{\eta_k} \subseteq U^*$ with $\|u - u_{\eta_k}\|_{\mathrm{C}^0(\Omega_U)} < \delta_k < \delta$ which implies

$$\left| F_*^{(k)}(u) - F_*^{(k)}(u_{\eta_k}) \right| < \frac{\varepsilon}{6L_1}.$$

By Lemma 2.5(1), $F_*^{(k)}(u_{\eta_k})$ is a continuous functional defined on the compact set U_{η_k} with dimension $n(\eta_k)$ and thus is equivalent to a continuous function (abusing notation) $F_*^{(k)} : \mathbb{R}^{n(\eta_k)} \to \mathbb{R}$. Therefore, by Theorem 2.4, we can find $M \in \mathbb{N}$, ξ_{kil}, θ_{ki} and $x_l \in \eta_k-$ net as defined in (3) on Ω_U, such that

$$\left| F_*^{(k)}(u_{\eta_k}) - \sum_{i=1}^{M} c_{ki}(F_*^{(k)}(\cdot))\sigma\left(\sum_{l=1}^{n(\eta_k)} \xi_{kil} u_{\eta_k}(x_l) + \theta_{ki} \right) \right| < \frac{\varepsilon}{6L_1}$$

which holds for all k. Setting the value $m = n(\eta_k)$ and since $u_{\eta_k}(x_l) = u(x_l)$ by (4), we have that

$$\begin{aligned}
&\left| F^{(k)}(u) - \sum_{i=1}^{M} c_{ki}(F_*^{(k)}(\cdot)) g\left(\sum_{l=1}^{m} \xi_{kil} u(x_l) + \theta_{ki}\right)\right| \\
&\leq \left|F^{(k)}(u) - F_*^{(k)}(u_{\eta_k})\right| + \left|F_*^{(k)}(u_{\eta_k}) - \sum_{i=1}^{M} c_{ki}(F_*^{(k)}(\cdot))\sigma\left(\sum_{l=1}^{m} \xi_{kil} u(x_l) + \theta_{ki}\right)\right| \\
&< \frac{\varepsilon}{3L_1},
\end{aligned}$$

where we also use the fact that $F^{(k)}(u) = F_*^{(k)}(u)$ since $u \in V$.

The extension operator E, from the Tietze extension theorem, is a continuous operator [14], and thus the coefficient of the expansions,

$$c_{ki}(F_*^k(\cdot)) = c_{ki}(E(c_k(F^k(\cdot)))) = c_{ki}(E(c_k(G[\alpha][\cdot])))),$$

are a continuous functionals depending on α. Hence, we can provide a similar argument for the approximation for $H^{(i,k)} : W \to \mathbb{R}$ defined by

$$H^{(k,i)}(\alpha) := c_{ki}(E(c_k(G[\alpha, \cdot])))).$$

By the Tietze extension theorem, we extend $H^{(i,k)}$ to a continuous functional $H_*^{(k,i)}$ on all of W^* (defined in Lemma 2.5) with $H_*^{(k,i)}(\alpha) = H^{(k,i)}(\alpha)$ for all $\alpha \in W$. Since W^* is a compact set, there exists $\delta' > 0$ such that

$$\left|H_*^{(k,i)}(\alpha_1) - H_*^{(k,i)}(\alpha_2)\right| < \frac{\varepsilon}{6L_2}$$

for all $\alpha_1, \alpha_2 \in \mathcal{A}^*$ with $\|\alpha_1 - \alpha_2\|_{C(\Omega_W)} < \delta'$, and

$$L_2 = \sum_{k=1}^{N}\sum_{i=1}^{M} \sup_{u\in U, x\in\Omega_V} \left|\sigma\left(\sum_{l=1}^{m} \xi_{kil} u(x_l) + \theta_{ki}\right) \cdot \sigma(\omega_k \cdot x + \zeta_k)\right|,$$

where L_2 is finite since the terms in the absolute value are continuous functions and the sets are compact. We can find an η_i-net defined on Ω_W, with $\delta_i' < \delta'$ and by Lemma 2.5 there exists $\alpha_{\eta_i} \in W_{\eta_i} \subseteq W^*$ (where $W_{\eta_i} = \{\alpha_{\eta_i} : \alpha \in W\}$) with $\|\alpha - \alpha_{\eta_i}\| < \delta_i < \delta$, which implies

$$\left|H_*^{(k,i)}(\alpha) - H_*^{(k,i)}(\alpha_{\eta_i})\right| < \frac{\varepsilon}{6L_2}.$$

The functionals $H_*^{(k,i)}(\alpha_{\eta_i})$ are equivalent to continuous functions defined on the compact set W_{η_i} of dimension $n(\eta_i)$, i.e., $H_*^{(k,i)} : \mathbb{R}^{n(\eta_i)} \to \mathbb{R}$. By Theorem 2.4, we can find $P \in \mathbb{N}$, $c_{kij}, \varphi_{kijh}, \rho_{kij}$ and $z_h \in \eta_i$- net defined on Ω_W, such that

$$\left|H_*^{(k,i)}(\alpha_{\eta_i}) - \sum_{j=1}^{P} c_{kij}\,\sigma\left(\sum_{h=1}^{n(\eta_i)} \varphi_{kijh}\alpha_{\eta_i}(z_h) + \rho_{kij}\right)\right| < \frac{\varepsilon}{6L_2},$$

which holds for all (k, i). Taking $p = n(\eta_i)$ and recalling that $\alpha_{\eta_i}(z_h) = \alpha(z_h)$, we have

$$\left|H_*^{(k,i)}(\alpha) - \sum_{j=1}^{P} c_{kij}\,\sigma\left(\sum_{h=1}^{p} \varphi_{kijh}\alpha(z_h) + \rho_{kij}\right)\right| < \frac{\varepsilon}{3L_2},$$

for all $\alpha \in W$.

Altogether, the following holds for all $(\alpha, u, y) \in W \times U \times \Omega_V$:

$$\left|G[\alpha][u](x) - \sum_{k=1}^{N}\sum_{i=1}^{M}\sum_{j=1}^{P} c_{kij}\,\sigma\left(\sum_{h=1}^{p} \varphi_{kijh}\alpha(z_h) + \rho_{kij}\right) \cdot \sigma\left(\sum_{l=1}^{m} \xi_{kil} u(x_l) + \theta_{ki}\right) \cdot \sigma(\omega_k \cdot x + \zeta_k)\right| < \varepsilon,$$

which concludes the proof.

□

If α is finite dimensional, then the proof simplifies and the following corollary holds.

Corollary 4.1. *Assume the same setting as in Theorem 3.4. Then, for any $\varepsilon > 0$, there exist a network as defined in Equation 6, such that*

$$|G[\alpha][u](y) - \mathrm{MONet}_{\mathrm{vect}}[\alpha][u](x)| < \varepsilon$$

holds for all $\alpha \in \Omega_W, u \in U$ and $x \in \Omega_V$.

4.2 Proof of Theorem 3.5

In this section, we prove the universal approximation property for the MNO and MONet networks in L^∞. We start with the next lemma which allows us to reformulate MONet with orthonormal components.

Lemma 4.2 (MONet network with orthonormal trunk and branch networks)**.** *Assume that Assumptions **A.2**, **S.1**, **S.2**, **S.3**, **M.1** and **M.2** hold. Then, we can re-write any* MONet *network defined in* (5) *as*

$$\mathrm{MONet}[\alpha][u](x) = \sum_{k=1}^{N} \sum_{\bar{\ell}=1}^{N \cdot M} \bar{\tau}_k(x) \bar{b}_{k\bar{\ell}}(u) \bar{L}_{k\bar{\ell}}(\alpha)$$

where $\{\bar{\tau}_k\}_{k=1}^N$ is a set of orthonormal neural networks with one hidden layer and a linear output layer with respect to the inner product in $\mathrm{L}^2_\lambda(\Omega_U)$, for $1 \leq k \leq N$, $\{\bar{b}_{k\bar{\ell}}\}_{\bar{\ell}=1}^{N \cdot M}$ is a set of orthonormal neural networks with one hidden layer and a linear output layer with respect to the $\mathrm{L}^2((U, \mu), \mathbb{R})$ inner product and $\{\bar{L}_{k\bar{\ell}}\}_{\bar{\ell}=1}^{N \cdot M}$ is a set of neural networks with one hidden layer and a linear output layer.

Proof. We first recall that a MONet network may be written as

$$\mathrm{MONet}[\alpha][u](x) = \sum_{k=1}^{N} \sum_{i=1}^{M} \tau_k(x) b_{ki}(u) L_{ki}(\alpha)$$

for $\tau_k(x) = \sigma(\omega_k \cdot x + \zeta_k)$, $b_{ki}(u) = \sigma\left(\sum_{l=1}^m \xi_{kil} u(x_l) + \theta_{ki}\right)$ and

$$L_{ki}(\alpha) = \sum_{j=1}^{P} c_{kij} \sigma \left(\sum_{h=1}^{p} \varphi_{kijh} \alpha(z_h) + \rho_{kij} \right).$$

We introduce the following notation:

- $\tau(x) \in \mathbb{R}^N$ the vector $\{\tau_k(x)\}_{k=1}^N$;
- $b_k(u) \in \mathbb{R}^M$ the vector $\{b_{ki}(u)\}_{i=1}^{i=M}$ for $1 \leq k \leq N$;
- $L_k(\alpha) \in \mathbb{R}^M$ the vector $\{L_{ki}(\alpha)\}_{i=1}^{i=M}$ for $1 \leq k \leq N$;
- $T(u, \alpha) \in \mathbb{R}^N$ the vector $\{\langle b_k(u), L_k(\alpha) \rangle_{\ell^2(\mathbb{R}^M)}\}_{k=1}^N$.

This allows us to re-write

$$\sum_{k=1}^{N} \sum_{i=1}^{M} \tau_k(x) b_{ki}(u) L_{ki}(\alpha) = \sum_{k=1}^{N} \tau_k(x) \langle b_k(u), L_k(\alpha) \rangle_{\ell^2(\mathbb{R}^M)} = \langle \tau(x), T(u, \alpha) \rangle_{\ell^2(\mathbb{R}^N)}.$$

The functions $\{\tau_k(x)\}_{k=1}^N$ are a finite set of $\mathrm{L}^2_\lambda(\Omega_U)$ functions by Assumptions **A.2** and **S.3**. Hence, by the Gram-Schmidt orthogonalization process, there exists an invertible matrix $\mathcal{Z} \in \mathbb{R}^{N \times N}$ such that $\mathcal{Z}\tau(y)$ is a set of orthogonal function with respect to the $\mathrm{L}^2_\lambda(K)$ inner product (otherwise, remove any terms from the summation that are redundant and reindex the summation). Let $Z = \mathcal{Z}^{-T}$ and we observe that

$$\langle \tau(x), T(u, \alpha) \rangle_{\ell^2(\mathbb{R}^N)} = \langle \mathcal{Z}\tau(y), ZT(u, \alpha) \rangle_{\ell^2(\mathbb{R}^N)}$$

$$= \sum_{k=1}^{N} [\mathcal{Z}\tau(x)]_k \, [ZT(u,\alpha)]_k \, .$$

We first note that the vector $\mathcal{Z}\tau(x) \in \mathbb{R}^N$ has entries

$$[\mathcal{Z}\tau(x)]_k = \sum_{r=1}^{N} [\mathcal{Z}]_{kr} \, \sigma(\omega_r \cdot x + \zeta_r)$$

which are neural networks with one hidden layer and a linear output layer. Second, we consider the vector $ZT(u,\alpha) \in \mathbb{R}^N$ which has entries

$$\begin{aligned}
[ZT(u,\alpha)]_k &= \sum_{r=1}^{N} [Z]_{kr} \, \langle b_r(u), L_r(\alpha) \rangle_{\ell^2(\mathbb{R}^M)} \\
&= \sum_{r=1}^{N} \sum_{s=1}^{M} [Z]_{kr} \, b_{rs}(u) L_{rs}(u) \\
&= \sum_{r=1}^{N} \sum_{s=1}^{M} \sum_{j=1}^{P} [Z]_{kr} \, \sigma\left(\sum_{l=1}^{m} \xi_{rsl} u(x_l) + \theta_{rs} \right) \sigma\left(\sum_{h=1}^{p} \varphi_{rsjh} \alpha(z_h) + \rho_{rsj} \right) . \qquad (26)
\end{aligned}$$

We proceed to the following change of variables: we consider the index variable $1 \leq \bar{\ell} \leq N \cdot M$ and replace every occurrence of r by $\lfloor (\bar{\ell}-1)/M \rfloor + 1$ and every occurrence of s by $((\bar{\ell}-1) \mod M) + 1$ where for $n, m \in \mathbb{N}$, $n \mod m$ denotes the remainder of the integer division of n by m. This allows us to define

- $\left[\tilde{Z}\right]_{k,\bar{\ell}} = [Z]_{k,\lfloor (\bar{\ell}-1)/M \rfloor + 1}$
- $\tilde{\xi}_{\bar{\ell},l} = \xi_{\lfloor (\bar{\ell}-1)/M \rfloor + 1, ((\bar{\ell}-1) \mod M)+1, l}$
- $\tilde{\theta}_{\bar{\ell}} = \theta_{\lfloor (\bar{\ell}-1)/M \rfloor + 1, ((\bar{\ell}-1) \mod M)+1}$
- $\tilde{\varphi}_{\bar{\ell},j,h} = \varphi_{\lfloor (\bar{\ell}-1)/M \rfloor + 1, ((\bar{\ell}-1) \mod M)+1, j, h}$
- $\tilde{\rho}_{\bar{\ell},j} = \rho_{\lfloor (\bar{\ell}-1)/M \rfloor + 1, ((\bar{\ell}-1) \mod M)+1, j}$

and we can continue from (26):

$$\begin{aligned}
[ZT(u,\alpha)]_k &= \sum_{\bar{\ell}=1}^{N \cdot M} \sigma\left(\sum_{l=1}^{m} \tilde{\xi}_{\bar{\ell},l} u(x_l) + \tilde{\theta}_{\bar{\ell}} \right) \left(\sum_{j=1}^{P} \left[\tilde{Z}\right]_{k,\bar{\ell}} \sigma\left(\sum_{h=1}^{p} \tilde{\varphi}_{\bar{\ell},j,h} \alpha(z_h) + \tilde{\rho}_{\bar{\ell},j} \right) \right) \\
&=: \langle \tilde{b}_k(u), \tilde{L}_k(\alpha) \rangle_{\ell^2(\mathbb{R}^{N \cdot M})}
\end{aligned}$$

where, for $1 \leq k \leq N$, $\tilde{b}_k(u) = \left\{ \tilde{b}_{k\bar{\ell}}(u) \right\}_{\bar{\ell}=1}^{N \cdot M} = \left\{ \sigma\left(\sum_{l=1}^{m} \tilde{\xi}_{\bar{\ell},l} u(x_l) + \tilde{\theta}_{\bar{\ell}} \right) \right\}_{\bar{\ell}=1}^{N \cdot M}$ and

$$\tilde{L}_k(\alpha) = \left\{ \tilde{L}_{k\bar{\ell}}(\alpha) \right\}_{\bar{\ell}=1}^{N \cdot M} = \left\{ \sum_{j=1}^{P} \left[\tilde{Z}\right]_{k,\bar{\ell}} \sigma\left(\sum_{h=1}^{p} \tilde{\varphi}_{\bar{\ell},j,h} \alpha(z_h) + \tilde{\rho}_{\bar{\ell},j} \right) \right\}_{\bar{\ell}=1}^{N \cdot M} .$$

We note that, for $1 \leq k \leq N$, $\tilde{b}_k(u)$ is a set of neural network with one hidden layer while $\tilde{L}_k(\alpha)$ is a set of neural networks with one hidden layer and one linear output layer.

By defining the orthonormal set of functions $\bar{\tau}(x) = \mathcal{Z}\tau(x)$, we obtain that

$$\mathrm{MONet}[\alpha][u](x) = \langle \tau(x), T(u,\alpha) \rangle_{\ell^2(\mathbb{R}^N)} = \sum_{k=1}^{N} \bar{\tau}_k(x) \langle \tilde{b}_k(u), \tilde{L}_k(\alpha) \rangle_{\ell^2(\mathbb{R}^{N \cdot M})} . \qquad (27)$$

For $1 \leq k \leq N$ and $1 \leq \bar{\ell} \leq N \cdot M$, every functional $\tilde{b}_{k\bar{\ell}}$ can be considered as a random variable mapping from the measure space (U, μ) into $(\mathbb{R}, \lambda)$. In particular, we have:

$$\int_U \tilde{b}_{k\bar{\ell}}(u)^2 \, \mathrm{d}\mu(u) = \int_U \sigma\left(\sum_{l=1}^{m} \tilde{\xi}_{\bar{\ell},l} u(x_l) + \tilde{\theta}_{\bar{\ell}}\right)^2 \mathrm{d}\mu(u) \leq \|\sigma\|^2_{\mathrm{L}^\infty(\mathbb{R})} \mu(U)$$

and the latter is finite by Assumptions **A.2** and **M.2**. By [3, p.6], $\tilde{b}_{k\bar{\ell}}(u)$ is therefore in the Hilbert space $\mathrm{L}^2((U,\mu),\mathbb{R})$ endowed with the inner product

$$\langle f, g \rangle_{\mathrm{L}^2((U,\mu),\mathbb{R})} = \int_U f(u) g(u) \, \mathrm{d}\mu(u)$$

for $f, g \in \mathrm{L}^2((U,\mu),\mathbb{R})$.

For $1 \leq k \leq N$, the functionals $\left\{\tilde{b}_{k\bar{\ell}}(u)\right\}_{\bar{\ell}=1}^{N\cdot M}$ are a finite set of $\mathrm{L}^2((U,\mu),\mathbb{R})$ functionals. By the Gram-Schmidt orthogonalization process, there therefore exists an invertible matrix $\mathcal{Z}_k \in \mathbb{R}^{(N\cdot M)\times(N\cdot M)}$ such that $\mathcal{Z}_k \tilde{b}_k(u)$ is a set of orthogonal functionals with respect to the $\mathrm{L}^2((U,\mu),\mathbb{R})$ inner product. Continuing from (27) and defining $Z_k = \mathcal{Z}_k^{-T}$, we have:

$$\mathrm{MONet}[\alpha][u](y) = \sum_{k=1}^{N} \bar{\tau}_k(x) \langle \mathcal{Z}_k \tilde{b}_k(u), Z_k \tilde{L}_k(\alpha) \rangle_{\ell^2(\mathbb{R}^{N\cdot M})}. \tag{28}$$

Similarly to the above, for $1 \leq k \leq N$, the vector $\mathcal{Z}_k \tilde{b}_k(u)$ has entries

$$\left[\mathcal{Z}_k \tilde{b}_k(u)\right]_{\bar{\ell}} = \sum_{r=1}^{N\cdot M} [\mathcal{Z}_k]_{\bar{\ell}r} \, \tilde{b}_{kr}(u) = \sum_{r=1}^{N\cdot M} [\mathcal{Z}_k]_{\bar{\ell}r} \, \sigma\left(\sum_{l=1}^{m} \tilde{\xi}_{r,l} u(x_l) + \tilde{\theta}_r\right)$$

which implies that $\mathcal{Z}_k \tilde{b}_k(u)$ is a set of neural networks with one hidden layer and a linear output layer. Furthermore, the vector $Z_k \tilde{L}_k(\alpha)$ has entries

$$\left[Z_k \tilde{L}_k(\alpha)\right]_{\bar{\ell}} = \sum_{r=1}^{N\cdot M} [Z_k]_{\bar{\ell},r} \, \tilde{L}_{k,r}(\alpha) = \sum_{r=1}^{N\cdot M} \sum_{j=1}^{P} [Z_k]_{\bar{\ell},r} \left[\tilde{Z}\right]_{k,r} \sigma\left(\sum_{h=1}^{p} \tilde{\varphi}_{rjh} \alpha(z_h) + \tilde{\rho}_{rj}\right). \tag{29}$$

We proceed to the following change of variables: we consider the index variable $1 \leq \bar{s} \leq N \cdot M \cdot P$ and replace every occurrence of r by $\left\lfloor \frac{\bar{s}-1}{P} \right\rfloor + 1$ and every occurrence of j by $((\bar{s}-1) \mod P) + 1$. This leads us to define the following variables

- $\left[\bar{Z}_k\right]_{\bar{\ell},\bar{s}} = [Z_k]_{\bar{\ell},\left\lfloor \frac{\bar{s}-1}{P} \right\rfloor + 1}$
- $\left[\bar{Z}\right]_{k,\bar{s}} = \left[\tilde{Z}\right]_{k,\left\lfloor \frac{\bar{s}-1}{P} \right\rfloor + 1}$
- $\bar{\varphi}_{\bar{s},h} = \tilde{\varphi}_{\left\lfloor \frac{\bar{s}-1}{P} \right\rfloor + 1, ((\bar{s}-1) \mod P)+1, h}$
- $\bar{\rho}_{\bar{s}} = \tilde{\rho}_{\left\lfloor \frac{\bar{s}-1}{P} \right\rfloor + 1, ((\bar{s}-1) \mod P)+1}$

and we can continue from (29):

$$\left[Z_k \tilde{L}_k(\alpha)\right]_{\bar{\ell}} = \sum_{\bar{s}=1}^{N\cdot M\cdot P} \left[\bar{Z}_k\right]_{\bar{\ell},\bar{s}} \left[\bar{Z}\right]_{k,\bar{s}} \sigma\left(\sum_{h=1}^{p} \bar{\varphi}_{\bar{s},h} \alpha(z_h) + \bar{\rho}_{\bar{s}}\right).$$

We note that $Z_k \tilde{L}_k(\alpha)$ is therefore a set of neural networks with one hidden layer and a linear output layer. For $1 \leq k \leq N$, by defining $\bar{b}_k(u) = \mathcal{Z}_k \tilde{b}_k(u) =: \{\bar{b}_{k\bar{\ell}}(u)\}_{\bar{\ell}=1}^{N\cdot M}$ and $\bar{L}_k(\alpha) = Z_k \tilde{L}_k(\alpha) =: \{\bar{L}_{k\bar{\ell}}(\alpha)\}_{\bar{\ell}=1}^{N\cdot M}$, we obtain from (28) that

$$\mathrm{MONet}[\alpha][u](x) = \sum_{k=1}^{N} \bar{\tau}(x) \langle \bar{b}_k(u), \bar{L}_k(\alpha) \rangle_{\ell^2(\mathbb{R}^{N\cdot M})}$$

$$= \sum_{k=1}^{N} \sum_{\bar{\ell}=1}^{N \cdot M} \bar{\tau}_k(x) \bar{b}_{k\bar{\ell}}(u) \bar{L}_{k\bar{\ell}}(\alpha)$$

where $\{\bar{\tau}_k\}_{k=1}^{N}$ is a set of orthonormal neural networks with one hidden layer and a linear output layer with respect to the inner product in $\mathrm{L}^2_\lambda(\Omega_V)$, for $1 \leq k \leq N$, $\{\bar{b}_{k\bar{\ell}}\}_{\bar{\ell}=1}^{N \cdot M}$ is a set of orthonormal neural networks with one hidden layer and a linear output layer with respect to the $\mathrm{L}^2((U,\mu),\mathbb{R})$ inner product and $\{\bar{L}_{k\bar{\ell}}\}_{\bar{\ell}=1}^{N \cdot M}$ is a set of neural networks with one hidden layer and a linear output layer.

□

The proof of Theorem 3.5 follows a classical idea in approximation theory. The key step is to approximate measurable mappings by continuous ones, which is possible on a arbitrarily large subset by Lusin's theorem [5, Theorem 7.1.13]. On this subset we apply Theorem 3.4, while on the remaining small complement the error is controlled via a clipping argument.

Proof of Theorem 3.5. Using the same argument as in the proof of Theorem 3.4, without loss of generality, we present the proof using the MONet architecture.

In the proof $C > 0$ will denote a constant that can be arbitrarily large, independent of all our parameters that may change from line to line.

For $M > 0$, let us define the truncated operator

$$G_M[\alpha] = \begin{cases} G[\alpha] & \text{if } \|G[\alpha]\|_{\mathrm{L}^2(\mu\times\lambda)(U\times\Omega_V)} \leq M \\ M \frac{G[\alpha]}{\|G[\alpha]\|_{\mathrm{L}^2(\mu\times\lambda)(U\times\Omega_V)}} & \text{else,} \end{cases}$$

from which we deduce that $\|G_M[\alpha]\|_{\mathrm{L}^2(\mu\times\lambda)(U\times\Omega_V)} \leq M$. We also note that for any function $\mathcal{N}[\alpha][u](x)$, we can upper bound the left-hand side of (8) as follows:

$$\begin{aligned} &\|G[\alpha][u](x) - \mathcal{N}[\alpha][u](x)\|_{\mathrm{L}^2_{\nu\times\mu\times\lambda}(W\times U\times\Omega_V)} \\ &\leq \|G[\alpha][u](x) - G_M[\alpha][u](x)\|_{\mathrm{L}^2_{\nu\times\mu\times\lambda}(W\times U\times\Omega_V)} + \|G_M[\alpha][u](x) - \mathcal{N}[\alpha][u](x)\|_{\mathrm{L}^2_{\nu\times\mu\times\lambda}(W\times U\times\Omega_V)} \\ &=: T_1 + T_2. \end{aligned} \tag{30}$$

We first show that $\lim_{M\to\infty} T_1 = 0$. In particular,

$$T_1^2 = \int_W \|G[\alpha][u](x) - G_M[\alpha][u](x)\|^2_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} \,\mathrm{d}\nu(\alpha)$$

and, for $\alpha \in W$, we note that:

$$\begin{aligned} \|G[\alpha][u](x) - G_M[\alpha][u](x)\|^2_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} &\leq C\left(\|G[\alpha][u](x)\|^2_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} + \|G_M[\alpha][u](x)\|^2_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)}\right) \\ &\leq C\|G[\alpha][u](x)\|^2_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} + M^2 \end{aligned} \tag{31}$$

where we used the fact that $\|G_M[\alpha][u](x)\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} \leq M$ in (31). Since

$$C\|G[\alpha][u](x)\|^2_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} + M^2 \in \mathrm{L}^1(\nu)$$

by Assumptions **M.1** and **O.3**, we can apply the dominated convergence to obtain:

$$\lim_{M\to\infty} T_1^2 = \int_W \lim_{M\to\infty} \|G[\alpha][u](x) - G_M[\alpha][u](x)\|^2_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} \,\mathrm{d}\nu(\alpha).$$

Now, for $\alpha \in W$,

$$\|G[\alpha][u](x) - G_M[\alpha][u](x)\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} \leq \|G[\alpha][u](x)\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} \mathbb{1}_{\{\|G[\alpha][u](x)\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} \geq M\}}$$

where $\mathbb{1}$ is the indicator function. By Assumption **O.3**, we have that $G[\alpha][u](x) \in \mathrm{L}^2_{\nu\times\mu\times\lambda}(W \times U \times \Omega_V)$ which implies that, ν-a.e., $\|G[\alpha][u](x)\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} < \infty$. Hence, ν-a.e.,

$$\begin{aligned}\lim_{M\to\infty} \|G[\alpha][u](x) - G_M[\alpha][u](x)\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} &\leq \lim_{M\to\infty} \|G[\alpha][u](x)\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} \mathbb{1}_{\{\|G[\alpha][u](x)\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} \geq M\}} \\ &= 0\end{aligned}$$

from which we deduce that $\lim_{M\to\infty} T_1 = 0$: we can therefore pick $M = M(\varepsilon/3) > \frac{2\varepsilon}{9\nu(W)^{1/2}}$ large enough so that

$$T_1 \leq \frac{\varepsilon}{3}. \tag{32}$$

We now tackle the T_2 term. We first note that the set $\mathrm{C}^0(U, V)$ is a Polish space by [28, Theorem 4.19] since U is a compact subset of the metric space $\mathrm{C}^0(\Omega_U)$ and V is Polish. By Assumptions **O.1** and **O.3**, this implies that $G_M : W \mapsto \mathrm{C}^0(U, V)$ is a Borel measurable map from a Polish space into another one. Define $\delta_1 = \frac{4\varepsilon}{(3M)^2}$: by [5, Theorem 7.1.13], we can therefore find a compact set $W_K \subseteq W$ with $\nu(W \setminus W_K) < \delta_1\varepsilon$ such that $G_M : W_K \mapsto \mathrm{C}^0(U, V)$ is continuous. Define $\delta_2 = 2(81\nu(W)\mu(U)\lambda(\Omega_V))^{-1/2}$: for the latter map, we can apply Theorem 3.4 to obtain a MONet network such that

$$\begin{aligned}&\sup_{\alpha\in W_K} \|G_M[\alpha][u](x) - \mathrm{MONet}[\alpha][u](x)\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} \\ &\leq \|G_M[\alpha][u](x) - \mathrm{MONet}[\alpha][u](x)\|_{\mathrm{L}^\infty_{\nu\times\mu\times\lambda}(W_K\times U\times\Omega_V)} (\mu(U)\lambda(\Omega_U))^{1/2} \\ &\leq \delta_2\varepsilon(\mu(U)\lambda(\Omega_U))^{1/2} = \frac{2\varepsilon}{9\nu(W)^{1/2}} \qquad (33)\end{aligned}$$

By Lemma 4.2, the MONet network may be re-written as

$$\mathrm{MONet}[\alpha][u](x) = \sum_{k=1}^{N}\sum_{i=1}^{M} \tau_k(x) b_{ki}(u) L_{ki}(\alpha)$$

where $\{\tau_k\}_{k=1}^N$ is a set of orthonormal neural networks with one hidden layer and a linear output layer with respect to the inner product in $\mathrm{L}^2_\lambda(\Omega_V)$, for $1 \leq k \leq N$, $\{b_{ki}\}_{i=1}^M$ is a set of orthonormal neural networks with one hidden layer and a linear output layer with respect to the $\mathrm{L}^2((U, \mu), \mathbb{R})$ inner product and $\{L_{ki}\}_{i=1}^M$ is a set of neural networks with one hidden layer and a linear output layer.

In particular, this implies that for all $\alpha \in W$,

$$\begin{aligned}\|\mathrm{MONet}[\alpha][u](x)\|^2_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} &= \sum_{k=1}^{N}\sum_{i=1}^{M}\sum_{l=1}^{N}\sum_{j=1}^{M} L_{ki}(\alpha)L_{lj}(\alpha) \int_U b_{ki}(u)b_{lj}(u)\,\mathrm{d}\mu(u) \int_{\Omega_V} \tau_k(x)\tau_l(x)\,\mathrm{d}\lambda(x) \\ &= \sum_{k=1}^{N}\sum_{i=1}^{M}\sum_{j=1}^{M} L_{ki}(\alpha)L_{kj}(\alpha) \int_U b_{ki}(u)b_{kj}(u)\,\mathrm{d}\mu(u) \qquad (34) \\ &= \sum_{k=1}^{N}\sum_{i=1}^{M} L_{ki}(\alpha)^2 \qquad (35) \\ &=: \|L(\alpha)\|^2_{\ell^2(\mathbb{R}^{N\cdot M})}\end{aligned}$$

where we used the orthonomality of $\{\tau_k\}_{k=1}^N$ for (34) and the fact that, for $1 \leq k \leq N$, $\{b_{k\bar{\ell}}\}_{\bar{\ell}=1}^M$ is a set of orthonormal functionals for (35).

Define $\delta_3 = \frac{2}{9\nu(W)^{1/2}}$ and the network

$$\overline{\mathrm{MONet}}[a][u](y) = \sum_{k=1}^{N}\sum_{i=1}^{M} \tau_k(x) b_{ki}(u) \gamma_{ki}(L_{ki}(\alpha))$$

where $\gamma(x) : \mathbb{R}^{N\cdot M} \mapsto \mathbb{R}^{N\cdot M}$ is a ReLU neural network with coordinates $\{\gamma_{ki}\}_{k=1,i=1}^{k=N,i=M}$ such that

$$\begin{cases} \|\gamma(x) - x\|_{\ell^2(\mathbb{R}^{N\cdot M})} < \varepsilon\delta_3 & \text{if } \|x\|_{\ell^2(\mathbb{R}^{N\cdot M})} \leq M + \frac{2\varepsilon}{9\nu(W)^{1/2}} \\ \|\gamma(x)\|_{\ell^2(\mathbb{R}^{N\cdot M})} \leq 2M & \text{for all } x \in \mathbb{R}^{N\cdot M} \end{cases}$$

which exists due to [34, Lemma C.2] and the fact that $M > \frac{2\varepsilon}{9\nu(W)^{1/2}}$. We now estimate as follows, starting from the T_2 term in (30) where $\mathcal{N}[\alpha][u](x) = \overline{\mathrm{MONet}}[a][u](x)$:

$$\begin{aligned} T_2 &= \left\|G_M[\alpha][u](x) - \overline{\mathrm{MONet}}[a][u](x)\right\|_{\mathrm{L}^2_{\nu\times\mu\times\lambda}(W\times U\times\Omega_V)} \\ &\leq \left\|G_M[\alpha][u](x) - \overline{\mathrm{MONet}}[a][u](x)\right\|_{\mathrm{L}^2_{\nu\times\mu\times\lambda}(W_K\times U\times\Omega_V)} \\ &\quad + \left\|G_M[\alpha][u](x) - \overline{\mathrm{MONet}}[a][u](x)\right\|_{\mathrm{L}^2_{\nu\times\mu\times\lambda}((W\setminus W_K)\times U\times\Omega_V)} \\ &\leq \|G_M[\alpha][u](x) - \mathrm{MONet}[a][u](x)\|_{\mathrm{L}^2_{\nu\times\mu\times\lambda}(W_K\times U\times\Omega_V)} \\ &\quad + \left\|\overline{\mathrm{MONet}}[a][u](x) - \mathrm{MONet}[a][u](x)\right\|_{\mathrm{L}^2_{\nu\times\mu\times\lambda}(W_K\times U\times\Omega_V)} \\ &\quad + \left\|G_M[\alpha][u](x) - \overline{\mathrm{MONet}}[a][u](x)\right\|_{\mathrm{L}^2_{\nu\times\mu\times\lambda}((W\setminus W_K)\times U\times\Omega_V)} \\ &=: T_3 + T_4 + T_5. \end{aligned} \tag{36}$$

By (33), we have that

$$T_3 \leq \sup_{\alpha\in W_K} \|G_M[\alpha][u](x) - \mathrm{MONet}[\alpha][u](x)\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)}\, \nu(W)^{1/2} \leq \frac{2\varepsilon}{9}. \tag{37}$$

For T_4, we start by computing the following: for $\alpha \in W_K$,

$$\begin{aligned} \|\mathrm{MONet}[\alpha][u](x)\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} &= \left(\sum_{k=1}^{N}\sum_{i=1}^{M} L_{ki}(\alpha)^2\right)^{1/2} \\ &\leq \|G_M[\alpha][u](x) - \mathrm{MONet}[\alpha][u](x)\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} + \|G_M[\alpha][u](x)\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} \\ &\leq \frac{2\varepsilon}{9\nu(W)^{1/2}} + M \end{aligned} \tag{38}$$

where we used (33) for (38). Then, we estimate:

$$\begin{aligned} T_4 &\leq \sup_{\alpha\in W_k} \left\|\overline{\mathrm{MONet}}[a][u](x) - \mathrm{MONet}[a][u](x)\right\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)}\, \nu(W_K)^{1/2} \\ &\leq \nu(W)^{1/2} \sup_{\alpha\in W_K} \|L(\alpha) - \gamma(L(\alpha))\|_{\ell^2(\mathbb{R}^{N\cdot M})} && (39) \\ &\leq \nu(W)^{1/2} \sup_{\|x\|_{\ell^2(\mathbb{R}^{N\cdot M})} \leq M + \frac{2\varepsilon}{9\nu(W)^{1/2}}} \|x - \gamma(x)\|_{\ell^2(\mathbb{R}^{N\cdot M})} && (40) \\ &\leq \nu(W)^{1/2}\varepsilon\delta_3 = \frac{2\varepsilon}{9} && (41) \end{aligned}$$

where we used the same computation to obtain (35) for (39), (38) for (40) and the definition of γ for (41).

For T_5, we estimate as follows:

$$\begin{aligned} T_5 &\leq \nu(W\setminus W_K)^{1/2} \left(\sup_{\alpha\in W\setminus W_K} \|\overline{\mathrm{MONet}}[\alpha][u](x)\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)} + \sup_{\alpha\in W\setminus W_K} \|G[\alpha][u](x)\|_{\mathrm{L}^2_{\mu\times\lambda}(U\times\Omega_V)}\right) \\ &\leq \nu(W\setminus W_K)^{1/2} \left(\|\gamma(L(\alpha))\|_{\ell^2(\mathbb{R}^{N\cdot M})} + M\right) && (42) \\ &\leq 3M\nu(W\setminus W_K)^{1/2} && (43) \\ &\leq 3M\,(\varepsilon\delta_1)^{1/2} = \frac{2\varepsilon}{9} && (44) \end{aligned}$$

where we proceeded analogously to (35) for (42) and used the definition of γ for (43).

Combining our estimates (37), (41) and (44), by (36), we deduce that $T_2 \leq \frac{2}{3}\varepsilon$. Using (32) and (30) allows us to conclude that

$$\left\|G[\alpha][u](x) - \overline{\mathrm{MONet}}[\alpha][u](x)\right\|_{\mathrm{L}^2_{\nu\times\mu\times\lambda}(W\times U\times\Omega_V)} \leq \varepsilon. \qquad \square$$

4.3 Scaling laws proofs

4.3.1 Proof of Theorem 3.6

We first prove the functional approximation rate in L^∞. The intuition behind the proof parallels our discussion in Section 2.2.

Proof of Theorem 3.6. Let $\delta > 0$ and $\mathcal{C}_U = \{\mathcal{B}_\delta(c_m)\}_{m=1}^{n_{c_U}}$ be a finite cover of Ω_U by c_U Euclidean balls where c_U can be further estimated by [41, Corollary 2]. By [41, Lemma 1], there exists a $\mathrm{C}^\infty(\Omega_U) \subseteq \mathrm{C}^\infty(\mathcal{C}_U)$ partition of unity $\{\omega_m(x) : \Omega_U \mapsto \mathbb{R}\}_{m=1}^{n_{c_U}}$ subordinate to the cover $\mathcal{C}_U$. This allows us to consider a discrete-to-continuum lifting from $[-\beta_U, \beta_U]^{n_{c_U}}$ to $\mathrm{C}^\infty(\Omega_U)$: we define the mapping $I_{\mathcal{C}_U} : [-\beta_U, \beta_U]^{n_{c_U}} \mapsto \mathrm{C}^\infty(\Omega_U)$ by

$$I_{\mathcal{C}_U}[z](x) = \sum_{m=1}^{n_{c_U}} [z]_m \omega_m(x)$$

for all $z \in [-\beta_U, \beta_U]^{n_{c_U}}$ and $x \in \Omega_U$. Conversely, we can define a continuum-to-discrete projection $P_{\mathcal{C}_U} : \mathrm{C}^0(\Omega_U) \mapsto [-\beta_U, \beta_U]^{n_{c_U}}$ by $P_{\mathcal{C}_U}(z) = (z(c_1), \ldots, z(c_{n_{C_U}}))^\top$ for $z \in \mathrm{C}^0(\Omega_U)$.

We note the following point-wise error approximation for any $u \in U$ and $x \in \Omega_U$:

$$\begin{aligned} |u(x) - I_{\mathcal{C}_U}[P_{\mathcal{C}_U}(u)](x)| &\le \sum_{m=1}^{n_{c_U}} |u(x) - u(c_m)||\omega_m(x)| \\ &= \sum_{m:\|x-c_m\|_2 \le \delta} |u(x) - u(c_m)||\omega_m(x)| \le L_U \delta \end{aligned}$$

implying that $\|u - I_{\mathcal{C}_U}[P_{\mathcal{C}_U}(u)]\|_{\mathrm{L}^\infty} \le \delta L_U$. Setting $\delta = \frac{\varepsilon}{2L_f L_U}$ and using the Lipschitz property of f, continuing from the above, we obtain

$$|f(u) - f(I_{\mathcal{C}_U}[P_{\mathcal{C}_U}(u)])| \le L_f \|u - I_{\mathcal{C}_U}[P_{\mathcal{C}_U}(u)]\|_{\mathrm{L}^\infty(\Omega_U)} \le L_f L_U \delta = \frac{\varepsilon}{2} \tag{45}$$

Next, we define $\hat{f} : [-\beta_U, \beta_U]^{n_{c_U}} \to \mathbb{R}$ such that $\hat{f}(z) = f(I_{\mathcal{C}_U}[z]) = f\left(\sum_{m=1}^{n_{c_U}} [z]_m \omega_m(x)\right)$. We claim that $\hat{f}$ is a Lipschitz function on $[-\beta_U, \beta_U]^{n_{c_U}}$. Indeed, let $z_1, z_2 \in [-\beta_U, \beta_U]^{n_{c_U}}$ and estimate as follows:

$$\begin{aligned} |\hat{f}(z_1) - \hat{f}(z_2)| =& |f(I_{\mathcal{C}_U}[z_1]) - f(I_{\mathcal{C}_U}[z_2])| \\ \le& L_f \|I_{\mathcal{C}_U}[z_1] - I_{\mathcal{C}_U}[z_2]\|_{\mathrm{L}^\infty(\Omega_U)} \\ \le& L_f \sup_{x\in\Omega_U} \sum_{m=1}^{n_{c_U}} |([z_1]_m - [z_2]_m)\,\omega_m(x)| \\ &\le L_f \sup_{x\in\Omega_U} \sqrt{\sum_{m=1}^{n_{c_U}} ([z_1]_m - [z_2]_m)^2} \sqrt{\sum_{m=1}^{n_{c_U}} (\omega_m(x))^2} \\ &\le L_f \|z_1 - z_2\|_{\ell^2(\mathbb{R}^{n_{c_U}})} \sup_{x\in\Omega_U} \sqrt{\sum_{m=1}^{n_{c_U}} \omega_m(x) \mathrm{d}x} \\ &= L_f \|z_1 - z_2\|_{\ell^2(\mathbb{R}^{n_{c_U}})}. \end{aligned}$$

Since $\hat{f}$ is Lipschitz continuous on the compact set $[-\beta_U, \beta_U]^{n_{c_U}}$, it is bounded by some constant $C_{\hat{f}}$ and we can deduce that $\hat{f} \in V(n_{c_U}, \beta_U, L_f, C_{\hat{f}})$ for some set of functions V (see Assumption **S.4**).

Consequently, we apply the function approximation Theorem 2.7. Specifically, for any $\varepsilon_0 > 0$, there exists a constant C depending on β_U and L_f such that the following holds. There exists

- a network class $\mathcal{F}_{\mathrm{NN}}(n_{c_U}, 1, L, p, K, \kappa, R)$ whose parameters scale as

  $$L = \mathcal{O}\left(n_{c_U}^2 \log(n_{c_U}) + n_{c_U}^2 \log(\varepsilon_0^{-1})\right), \quad p = \mathcal{O}(1), \quad K = \mathcal{O}\left(n_{c_U}^2 \log n_{c_U} + n_{c_U}^2 \log(\varepsilon_0^{-1})\right),$$
  $$\kappa = \mathcal{O}(n_{c_U}^{n_{c_U}/2+1} \varepsilon_0^{-n_{c_U}-1}), \qquad R = 1$$

  where the constants hidden in $\mathcal{O}$ depend on β_U and L_f,

- networks $\{b_k\}_{k=1}^{H^{n_{c_U}}} \subset \mathcal{F}_{\mathrm{NN}}(n_{c_U}, 1, L, p, K, \kappa, R)$ with $H := C\sqrt{n_{c_U}}\varepsilon_0^{-1}$ and
- points $\{s_k\}_{k=1}^{H^{n_{c_U}}} \subset [-\beta_U, \beta_U]^{n_{c_U}}$

such that

$$\sup_{z\in[-\beta_U,\beta_U]^{n_{c_U}}} \left| \hat{f}(z) - \sum_{k=1}^{H^{n_{c_U}}} \hat{f}(s_k) b_k(z) \right| \leq \varepsilon_0.$$

We note that $P_{\mathcal{C}_U}(U) \subset [-\beta_U, \beta_U]^{n_{c_U}}$ by the fact that U satisfies $U(d_U, \gamma_U, L_U, \beta_U)$ and hence, setting $\varepsilon_0 = \varepsilon/2$, this does not change the network class scalings,

$$\sup_{u\in U} \left| \hat{f}(P_{\mathcal{C}_U}[u]) - \sum_{k=1}^{H^{n_{c_U}}} \hat{f}(s_k) b_k(P_{\mathcal{C}_U}[u]) \right| \leq \frac{\varepsilon}{2}. \tag{46}$$

We conclude as follows: using (45) and (46), for any $u \in U$, we have

$$\begin{aligned}
\sup_{u\in U} \left| f(u) - \sum_{k=1}^{H^{n_{c_U}}} \hat{f}(s_k) b_k(P_{\mathcal{C}_U}(u)) \right| &\leq \sup_{u\in U} \left| f(u) - \hat{f}(P_{\mathcal{C}_U}(u)) \right| \\
&\quad + \sup_{u\in U} \left| \hat{f}(P_{\mathcal{C}_U}(u)) - \sum_{k=1}^{H^{n_{c_U}}} \hat{f}(s_k) b_k(P_{\mathcal{C}_U}(u)) \right| \\
&= \sup_{u\in U} |f(u) - f(I_{\mathcal{C}_U}[P_{\mathcal{C}_U}(u)])| + \sup_{u\in U} \left| \hat{f}(P_{\mathcal{C}_U}[u]) - \sum_{k=1}^{H^{n_{c_U}}} \hat{f}(s_k) b_k(P_{\mathcal{C}_U}[u]) \right| \\
&\leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.
\end{aligned}$$

Recalling that $\hat{f}(s_k) = f(I_{\mathcal{C}_U}[s_k])$, we set $u_k = I_{\mathcal{C}_U}[s_k]$ and obtain the claim of the theorem. □

4.3.2 Proof of Theorem 3.16

In this section, we derive the convergence rate in the multiple operator setting. In particular, the proof is an application of Theorems 3.6 and 3.8.

Proof of Theorem 3.16. For $u \in U$ and $x \in \Omega_V$, define the functional $f_{u,x} : \{\alpha : \Omega_W \mapsto \mathbb{R} \,|\, \|\alpha\|_{\mathrm{L}^\infty} \leq \beta_W\} \mapsto \mathbb{R}$ as

$$f_{u,x}(\alpha) = G[\alpha][u](x).$$

In particular, we have that

$$\begin{aligned}
|f_{u,x}(\alpha_1) - f_{u,x}(\alpha_2)| &= |G[\alpha_1][u](x) - G[\alpha_2][u](x)| \\
&\leq L_G \|\alpha_1 - \alpha_2\|_{\mathrm{L}^{r_G}(\Omega_W)} \\
&\leq L_G |\Omega_W|^{1/(r_G)} \|\alpha_1 - \alpha_2\|_{\mathrm{L}^\infty(\Omega_U)}.
\end{aligned} \tag{47}$$

where we use (21) for (47). Therefore, we can apply Theorem 3.6. Specifically, for any $\varepsilon_0 > 0$, there exists constants C'' and C_ζ depending on $\beta_W, L_G|\Omega_W|^{1/r_G}$ and $L_G|\Omega_W|^{1/r_G}, L_W$ respectively such that the following holds. There exists

- a constant $\zeta := C_\zeta \varepsilon$ and points $\{y_m\}_{m=1}^{n_{c_W}} \subset \Omega_W$ so that $\{\mathcal{B}_\zeta(y_m)\}_{m=1}^{n_{c_W}}$ is a cover of Ω_W for some n_{c_W},
- a network class $\mathcal{F}_3 = \mathcal{F}_{\mathrm{NN}}(n_{c_W}, 1, L_3, p_3, K_3, \kappa_3, R_3)$ whose parameters scale as

$$\begin{aligned}
&L = \mathcal{O}\left(n_{c_W}^2 + n_{c_W} \log(\varepsilon_0^{-1})\right), \quad p = \mathcal{O}(1), \quad K = \mathcal{O}\left(n_{c_W}^2 \log n_{c_W} + n_{c_W}^2 \log(\varepsilon_0^{-1})\right), \\
&\kappa = \mathcal{O}(n_{c_W}^{n_{c_W}/2+1} \varepsilon_0^{-n_{c_W}-1}), \qquad R = 1
\end{aligned}$$

  where the constants hidden in $\mathcal{O}$ depend on β_W and $L_G|\Omega_W|^{1/r_G}$,

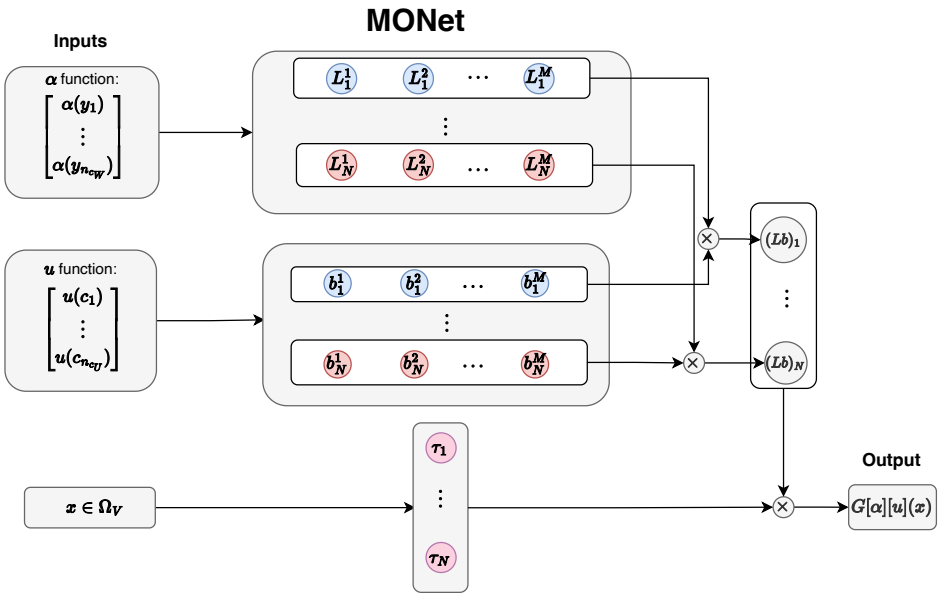

Figure 1: **MONet architecture**: The α function is the input for the parameter-approximation network. The u function is the input for the function-approximation network. The spatial values $x \in \Omega_V$ are the input for the space-approximation network.

- networks $\{l_p\}_{p=1}^{P^{n_{c_W}}} \subset \mathcal{F}_3$ with $P := C'' \sqrt{n_{c_W}} \varepsilon_0^{-1}$ and
- functions $\{\alpha_p\}_{p=1}^{P^{n_{c_W}}} \subset \{\alpha : \Omega_W \mapsto \mathbb{R} \,|\, \|\alpha\|_{\mathrm{L}^\infty} \leq \beta_W\}$

such that

$$\sup_{\alpha \in W} \left| f_{u,x}(\alpha) - \sum_{p=1}^{P^{n_{c_W}}} f_{u,x}(\alpha_k) l_p(P_{\mathcal{C}_W}(\alpha)) \right| = \sup_{\alpha \in U} \left| G[\alpha][u](x) - \sum_{p=1}^{P^{n_{c_W}}} G[\alpha_p][u](x) l_p(P_{\mathcal{C}_W}(\alpha)) \right| \leq \varepsilon_0$$

where $P_{\mathcal{C}_W}(\alpha)$ is defined in the proof of Theorem 3.6.

By assumption, $G[\alpha_p] \in \mathcal{G}$ for all $1 \leq p \leq P^{n_{c_W}}$. This corresponds to the situation in (14) where $G^{(p)} = G[\alpha_p]$, i.e. to the problem of approximating $P^{n_{c_W}}$ single operators. For $1 \leq p \leq P^{n_{c_W}}$, we therefore apply Theorem 3.8 and the rest of the argument to obtain (22) is analogous to the proof of Theorem 3.8 in Remarks 3.9 and 3.13 (with $\varepsilon_0 = \varepsilon/2$ and $\varepsilon_1 = \varepsilon/(2P^{n_{c_W}})$). □

5 Numerical Experiments

In this section, we refer to τ as the space-approximation network, b as the function-approximation network, and L or l as the parameter-approximation network. To evaluate the versatility and effectiveness of MNO and MONet, we test both architectures (see Figures 2 and 1) on five representative parametric PDEs, spanning settings in which the parameter α is modeled either as a function or as a finite-dimensional vector in $\mathbb{R}^p$. In all experiments, the objective is to predict the PDE solution at points $x = (t, x_{\text{spatial}}) \in (0, 2] \times [0, 2]$, given the parametric function α and the initial condition u_0.

We construct 50 initial conditions for each PDE following the sinusoidal formulation proposed in [56]:

$$u_0(x) = \sum_{i=1}^{4} A_i \sin(k_i x + \phi_i), \tag{48}$$

where $k_i = \pi n_i$ and n_i are uniformly sampled integers in $[1, 4]$. The amplitudes A_i are sampled uniformly from $[0, 1]$, and ϕ_i are random phases drawn from $(0, 2\pi)$. Following the setup in [54,56], after computing (48), each

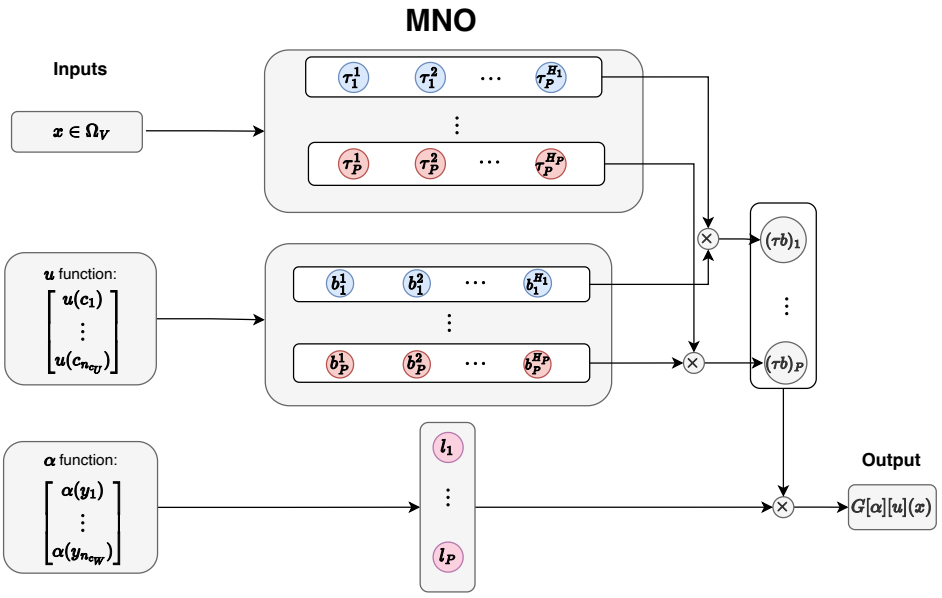

Figure 2: **MNO architecture**: The α function is the input for the parameter-approximation network. The u function is the input for the function-approximation network. The spatial values $x \in \Omega_Y$ are the input for the space-approximation network.

initial condition undergoes random post-processing: with 10% probability, the absolute value of u_0 is taken, with 50% probability, its sign is flipped (i.e., multiplied by -1), and with 10% probability, it is multiplied by the indicator function of a randomly chosen smooth subdomain of $[0, 2]$.

The resulting initial conditions are then sampled at points $\{c_i\}_{i=1}^{n_{c_U}}$ over the domain $[0, 2]$ and provided as inputs to the function-approximation networks. In particular, we use: $n_{c_U} = 64$ and cell-center points (i.e. midpoints of uniform grid cells) $\{c_i\}_{i=1}^{64}$. The space-approximation networks take as input the spatiotemporal coordinates $(t, x) \in \mathbb{R}^2$.

All models are trained with mean squared loss (MSE). We evaluate them on a 32×64 spacetime grid over $[0, 2] \times [0, 2]$, and report the average relative L^2 error across all test cases:

$$\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \frac{\|u_{\text{pred}}^{(i)} - u_{\text{target}}^{(i)}\|_2}{\|u_{\text{target}}^{(i)}\|_2 + \varepsilon},$$

where $(u_{\text{pred}}^{(i)}, u_{\text{target}}^{(i)})$ corresponds to the i-th pair of predicted and reference solution functions, each evaluated at all points of the discretized spacetime grid, $\varepsilon = 10^{-5}$ and $N_{\text{test}} = 80 \times 50$, corresponding to 80 distinct parameter samples α (i.e., 80 distinct parametrized PDEs), each evaluated with 50 different initial conditions u_0.

We compare our models MNO and MONet, with DeepONet and MIONet with different configurations, as detailed in Table 8. We employ two network configurations for MNO: MNO-S (small) which uses 1.19M parameters and MNO-L (large) which uses 16.7M. This is computational feasible since MNO's tensor structure is more amenable to larger model complexity. In "DeepONet-C", we simply concatenate the α and u inputs together and put them into a single function-approximation layer. This is also a theoretically valid approach to training multiple operators; however, as shown in the experiments, does not preform as well as MNO and MONet. Note that in the experiments, the training time for MNO-S, MONet, and DeepONet are comparable, while the training times for MNO-L and MIONet are larger as expected. Additional details on the experimental setup, including network hyperparameters and training times, are provided in Appendix A.

	DeepONet	DeepONet-C	MIONet	MONet	MNO-S	MNO-L
Number of parameters	1.47M	1.47M	1.50M	1.15M	1.19M	16.7M
# of L or l (P)		**N/A**		1	10	40
Depth of L or l		**N/A**		4	4	4
# of b (M/H)	100	100	75	100	20	100
Depth of b	4	4	4	4	4	4
# of τ (N)	20	20	75	20	20	20
Depth of τ	6	6	6	6	6	6

Table 8: Model configurations and architectural details for all tested variants. Abbreviations: **S** refers to the small version; **L** to the large version; and the symbol # denotes the number of corresponding elements. For reference, the number of parameter-approximation networks (L or l) corresponds to P in Equations (5) and (22); the number of function-approximation networks (b) corresponds to M in Equation (5) and H in Equation (22); and the number of space-approximation networks (τ) corresponds to N in both Equation (5) and (22). For simplicity, all powers of P, M, H, and N are omitted.

5.1 Conservation Laws

We consider the following one-dimensional conservation law with periodic boundary conditions:

$$\begin{aligned} u_t + (\alpha_1 u + \alpha_2 u^2 + \alpha_3 u^3)_x &= \alpha_4 u_{xx}, \quad (t,x) \in [0,2] \times [0,2], \\ u(0,x) &= u_0(x), \\ u(t,0) &= u(t,2), \end{aligned}$$

where the parameter vector $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3, \alpha_4]^\top$ is encoded within the parameter-approximation layers of both MNO and MONet. The components of $\boldsymbol{\alpha}$ are sampled from the ranges $\alpha_i \in [0.9\alpha_i^c, 1.1\alpha_i^c]$, with the reference values given by $\alpha^c = [1, 1, 1, 0.1]^\top$.

Table 9: Performance comparison on conservation laws. In in-distribution (IN) experiments, we set $\alpha_i \in [0.9\alpha_i^c, 1.1\alpha_i^c]$, and in out-of-distribution (OOD) experiments, we set $\alpha_i \in [0.8\alpha_i^c, 1.2\alpha_i^c]$

Model	**Relative L^2 Error**	
	IN	OOD
DeepONet	6.59%	9.00%
DeepONet-C	5.36%	6.59%
MIONet	5.65%	8.48%
MONet	5.67%	7.20%
MNO-S	4.49%	6.64%
MNO-L	**3.84%**	**5.92%**

In this experiment, the family of operators emit solutions with similar (viscous) shock or rarefraction profiles, mainly differing in speeds. The space of potential solutions likely lie on a lower dimensional structure which shares commonalities between each randomly sampled PDE (i.e. each randomized flux). Thus we expect that the empirical rates and scalings are faster than the general rates proven in the previous sections. We observe that MNO and MONet outperform DeepONet-type architectures with comparable parameter counts, demonstrating the efficacy of its α-encoding strategy over simple concatenation in the function-approximation networks. Our smaller networks produce in-distribution and out-of-distribution errors which are lower than the standard and concatenated DeepONet (see Table 9). Figure 3 shows that the (local) errors for DeepONet and MIONet are more concentrated in the shock formation and dynamics, while MNO and MONet demonstrate a more even error distribution with relatively less error around regions of large gradients.

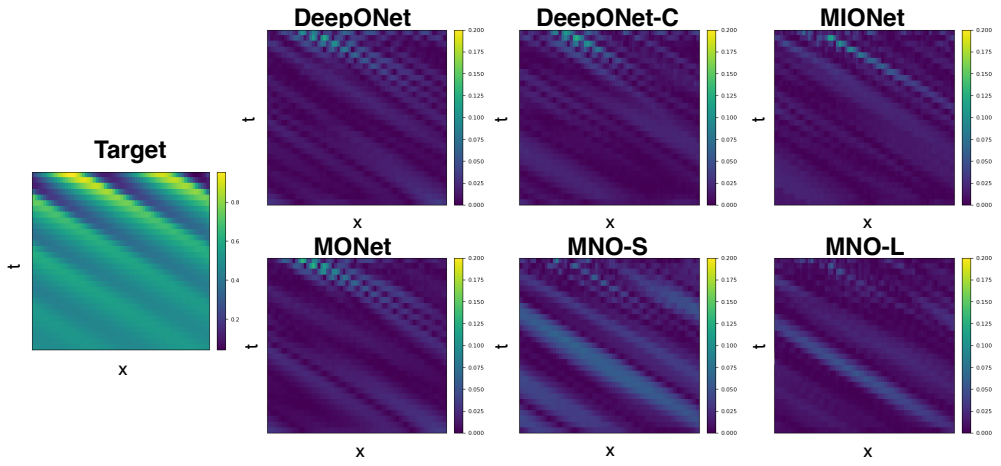

Figure 3: **Representative solution for conservation laws**: The target solution (left) and error maps for DeepONet, DeepONet-C, MIONet, MONet, MNO-S and MNO-L. The instance-specific relative errors are 5.49%, 4.82%, 2.92%, 5.11%, 2.52% and 1.81%, respectively, aligning with the trends observed in Table 9.

5.2 Diffusion-Reaction-Advection Equation

We consider the following one-dimensional diffusion–reaction–advection equation:

$$\begin{aligned} u_t &= \alpha_1 u_{xx} + \alpha_2 u_x + \alpha_3 u^{\alpha_4}(1 - u^{\alpha_5}), \quad (t, x) \in [0, 2] \times [0, 2], \\ u(0, x) &= u_0(x), \\ u(t, 0) &= u(t, 2), \end{aligned}$$

where the parameter vector $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5]^\top$ is encoded within the parameter-approximation layers of both MNO and MONet. The first three components are sampled from the ranges $\alpha_i \in [0.9\alpha_i^c, 1.1\alpha_i^c]$ for $1 \leq i \leq 3$, with reference values $\alpha^c = [0.01, 1, 1]^\top$, while α_4 and α_5 are drawn uniformly from $[1, 3]$.

In this experiment, the parameters change the governing equation in a nonlinear fashion. This may be the cause for the larger error observed in the DeepONet models. From Table 10, we observe that MNO and MONet variants produce more accurate solutions even with comparable parameter counts. The MNO and MONet produce better in- and out-of-distribution predictions. The structured encoding in MNO ensures more effective parameter sharing, which could be contributing to the lower error rates. Figure 4 illustrates this performance difference: notably, MNO is able to substantially reduce errors in regions that consistently show elevated error across all other methods. This shows an intrinsic difference between the underlying features learned by the family of models.

5.3 Nonlinear Klein-Gordon Equation

We consider the following nonlinear Klein–Gordon equation:

$$u_{tt} = \alpha_1^2 u_{xx} - \alpha_2^2 \alpha_1^4 u - \alpha_3 u^3, \quad (t, x) \in [0, 2] \times [0, 2],$$

Table 10: Performance comparison on diffusion-reaction-advection equation. For $1 \leq i \leq 3$, in in-distribution (IN) experiments, $\alpha_i \in [0.9\alpha_i^c, 1.1\alpha_i^c]$ whereas in out-of-distribution (OOD) experiments, we set $\alpha_i \in [0.8\alpha_i^c, 1.2\alpha_i^c]$.

Model	**Relative L^2 Error**	
	IN	OOD
DeepONet	13.63%	15.10%
DeepONet-C	4.91%	7.07%
MIONet	3.95%	7.06%
MONet	3.80%	6.21%
MNO-S	3.39%	5.47%
MNO-L	**2.51%**	**4.27%**

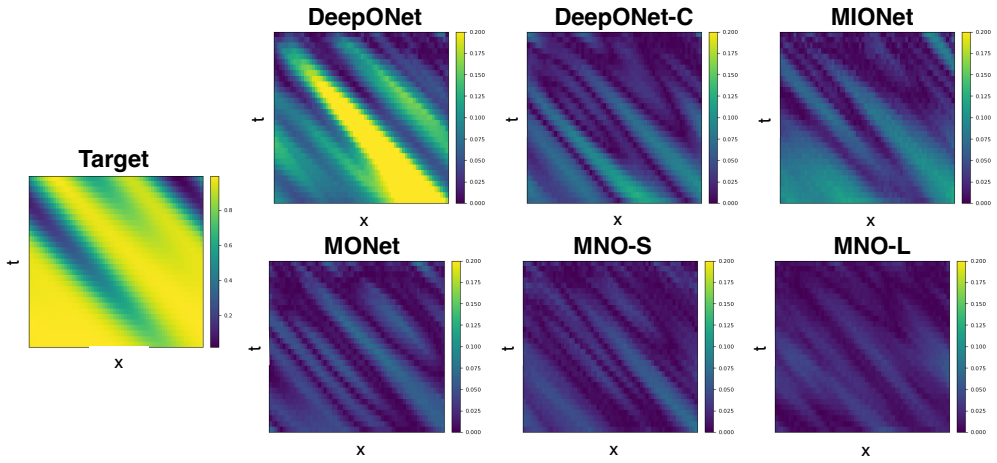

Figure 4: **Representative solution for diffusion-reaction-advection equation**: The target solution (left) and error maps for DeepONet, DeepONet-C, MIONet, MONet, MNO, and MNO-L. The instance-specific relative errors are 15.02%, 5.26% , 6.27% , 5.26%, 3.08% and 2.38%, respectively, aligning with the trends observed in Table 10.

$$
\begin{aligned}
u(0,x) &= u_0(x), \\
u_t(0,x) &= 0, \\
u(t,0) &= u(t,2).
\end{aligned}
$$

The parameter vector $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3]^\top$ is encoded within the parameter-approximation layers of both MNO and MONet. The components of $\boldsymbol{\alpha}$ are sampled from the ranges $\alpha_i \in [0.9, \alpha_i^c, , 1.1, \alpha_i^c]$ with reference values $\alpha^c = [1, 1, 1]^\top$.

Table 11: Performance comparison on the nonlinear Klein-Gordon equation. In in-distribution (IN) experiments, $\alpha_i \in [0.9\alpha_i^c, 1.1\alpha_i^c]$ whereas in out-of-distribution (OOD) experiments, we set $\alpha_i \in [0.85\alpha_i^c, 1.15\alpha_i^c]$ for $i \in [1, 2, 3]$

Model	**Relative L^2 Error**	
	IN	OOD
DeepONet	24.03%	33.82%
DeepONet-C	5.67%	7.90%
MIONet	7.73%	13.78%
MONet	4.53%	7.87%
MNO-S	3.56%	7.30%
MNO-L	**2.50%**	**5.90%**

In this experiment, the governing equation is a second-order hyperbolic PDE and thus produces wave-like solutions. Notably, Table 11 shows that DeepONet-C achieves lower relative errors than MIONet on this task, while MNO further improves performance, yielding substantially smaller errors overall. Figure 5 shows that most models' errors have coarse and low-frequency patterns appear while MNO does not. Additionally, as the parameter counts increase, the error associated with MNO decreases locally as well.

5.4 Parametric Diffusion-Reaction Equation

We consider the following parametric diffusion–reaction equation:

$$
\begin{aligned}
u_t &= (\alpha(x)u_x)_x + u(1-u), \quad (t,x) \in [0,2] \times [0,2], \\
u(0,x) &= u_0(x), \\
u(t,0) &= u(t,2),
\end{aligned}
$$

where the spatially varying diffusivity $\alpha(x)$ is sampled from a Gaussian random process with variance 0.01^2. The parametric function $\alpha(x)$ is evaluated at 129 sensor locations corresponding to the boundaries of uniformly

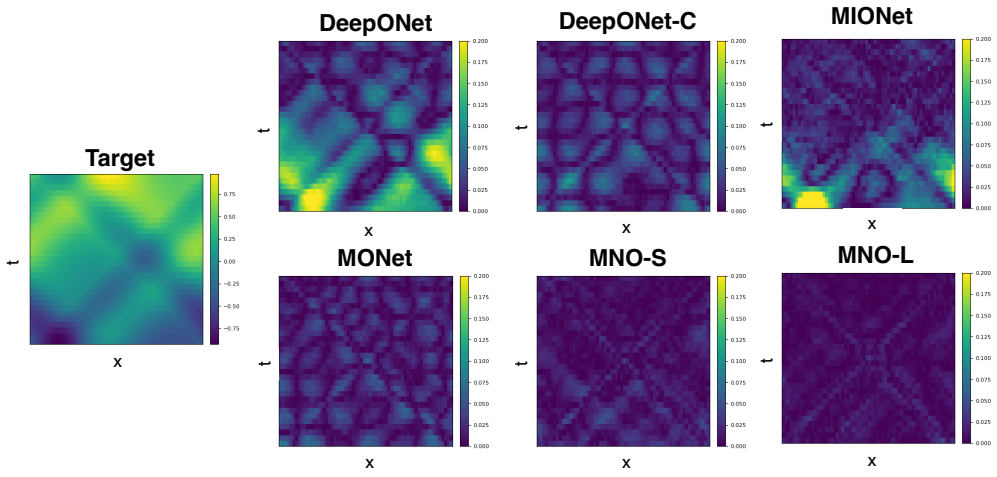

Figure 5: **Representative solution for the nonlinear Klein-Gordon equation**: The target solution (left) and error maps for DeepONet, DeepONet-C, MIONet, MONet, MNO-S, and MNO-L. The instance-specific relative errors are 19.84%, 7.99%, 14.26%, 6.06%, 3.76% and 2.34%, respectively, aligning with the trends observed in Table 11.

spaced cells, $\{x_i^b\}_{i=1}^{129}$, and the resulting values $\alpha(x_i^b)$ are encoded within the parameter-approximation networks of MNO and MONet.

This problem is more challenging since the parametric inputs are spatial dependent and are differentiated within the diffusion term. As shown in Table 12, MNO-L achieves the highest accuracy among all tested models. This improvement stems from the structured parameter encoding introduced by the parameter-approximation layers, yielding substantially better performance than simply concatenating α with the function-approximation inputs. Figure 6 further illustrates that, although all models exhibit localized error regions near the bottom right of the domain, MNO markedly reduces this region and yields significantly lower local errors.

5.5 Parametric wave Equation

We consider the following parametric wave equation:

$$\begin{aligned} u_{tt} &= \alpha^2(t) u_{xx}, \quad (t,x) \in [0,2] \times [0,2], \\ u(0,x) &= u_0(x), \\ u_t(0,x) &= 0, \\ u(t,0) &= u(t,2), \end{aligned}$$

where the time-dependent parameter function $\alpha(t)$ is drawn from a Gaussian random process with variance 1. The parametric function $\alpha(t)$ is evaluated at 64 sensor locations corresponding to the boundaries of uniformly spaced cells, $\{t_i^b\}_{i=1}^{64}$, and the resulting values $\alpha(t_i^b)$ are encoded within the parameter-approximation networks of MNO and MONet.

Since the parametric function is the time-dependent wave speed, an error in capturing the dependence can lead to incorrect dynamics for all time. From Figure 7 we see that MNO and MONet demonstrate a more balanced and overall lower local error distribution (see also Table 13), whereas the remaining models

Table 12: Performance comparison on the parametric diffusion-reaction equation (in-distribution).

Model	**Relative L^2 Error**
DeepONet	9.68%
DeepONet-C	6.59%
MIONet	5.65%
MONet	5.77%
MNO-S	4.62%
MNO-L	**3.34%**

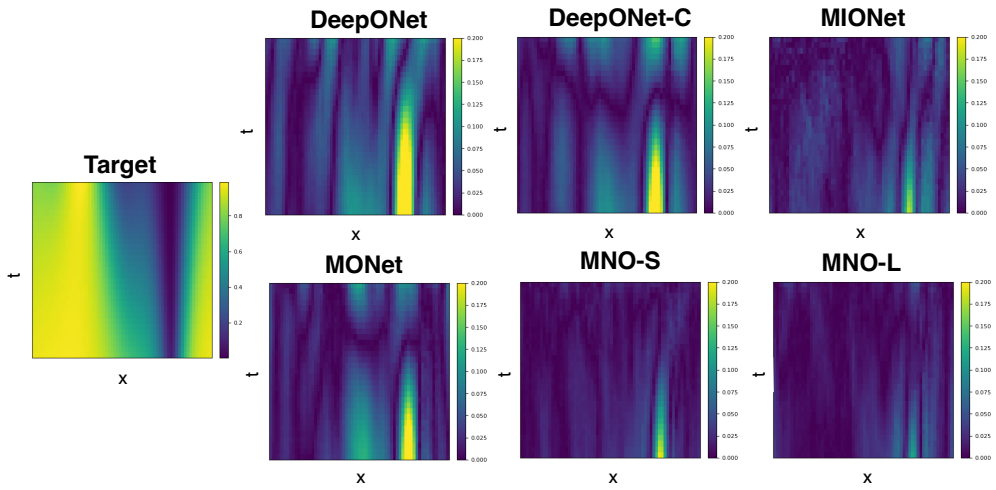

Figure 6: **Representative solution for the parametric diffusion-reaction equation**: The target solution (left) and error maps for DeepONet, DeepONet-C, MIONet, MONet,MNO-S and MNO-L. The instance-specific relative errors are 9.66%, 7.52%, 4.60%, 6.84%, 3.48% and 2.83%, respectively, aligning with the trends observed in Table 12.

Table 13: Performance comparison on the parametric diffusion-reaction equation (in-distribution).

Model	**Relative L^2 Error**
DeepONet	56.37%
DeepONet-C	9.31%
MIONet	13.66%
MONet	6.95%
MNO-S	5.72%
MNO-L	**4.41**%

show pronounced error concentrations and patterned error. The patterns likely indicate that larger features are missing in the model. In particular, in regions with higher contrast, the comparable models emit coarse scale errors that degrade their predictive capabilities.

6 Conclusion

In this work, we provided theoretical insights into the problem of learning a collection of operators using neural networks. For the multiple operator learning setting, we introduced two new architectures, MNO and MONet, and established their universal approximation properties across different classes of operators. Our analysis covered continuous, integrable, and Lipschitz operators. In the latter case, we derived explicit scaling laws for MNO, quantifying how the network size must grow to achieve a prescribed approximation accuracy. We further empirically validated the effectiveness of both architectures on a range of parametric PDE problems, confirming their strong performance in practice. For the case of learning several single operators, we showed that the theoretical approximation order yields new insights into how computational complexity can be balanced among subnetworks and how overall scaling efficiency can be improved. This provides a principled framework for architectural design.

Future research directions in the multiple operator learning context include establishing lower bounds on approximation and sample complexity similarly to [35], developing a rigorous theory of generalization error as in [41], extending the current analysis of approximation order, and exploring possible extensions to kernel-based operator learning frameworks [2, 23].

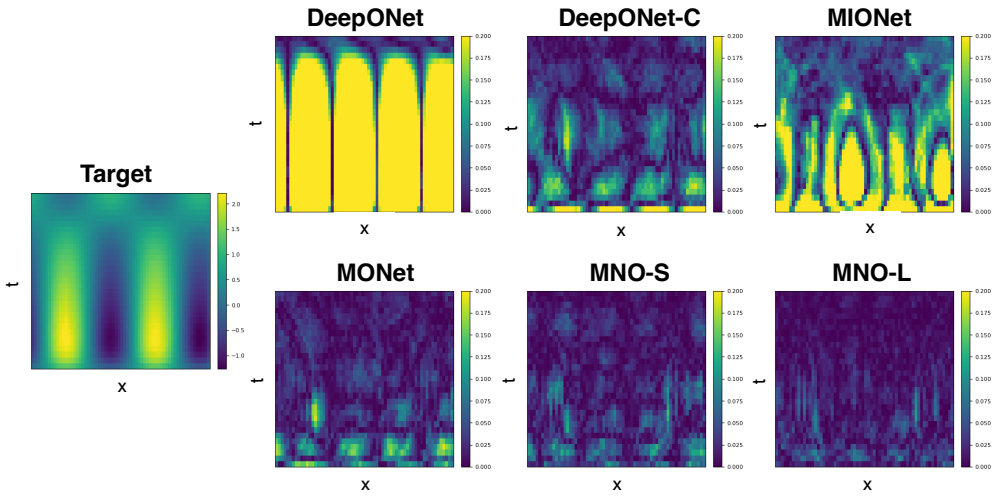

Figure 7: **Representative solution for the parametric wave equation**: The target solution (left) and error maps for DeepONet, DeepONet-C, MIONet, MONet, MNO-S and MNO-L, The instance-specific relative errors are 79.45%, 6.99%, 17.33% , 4.83%, 3.70% and 2.47%, respectively, aligning with the trends observed in Table 12.

Acknowledgment

AW and HS were supported in part by NSF DMS 2427558. JS was supported in part by AFOSR FA9550-23-1-0445. ZZ was supported by the U.S. Department of Energy (DOE) Office of Science Advanced Scientific Computing Research program DE-SC0025440.

References

[1] Anima Anandkumar, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Nikola Kovachki, Zongyi Li, Burigede Liu, and Andrew Stuart. Neural operator: Graph kernel network for partial differential equations. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2019.

[2] Pau Batlle, Matthieu Darcy, Bamdad Hosseini, and Houman Owhadi. Kernel methods are competitive for operator learning. *Journal of Computational Physics*, 496:112549, 2024.

[3] Alain Berlinet and Christine Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer, New York, NY, 2004.

[4] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B. Kovachki, and Andrew M. Stuart. Model Reduction And Neural Networks For Parametric PDEs. *The SMAI Journal of computational mathematics*, 7:121–157, 2021.

[5] Vladimir I. Bogachev. *Measure theory*. Springer Berlin, Heidelberg, Berlin, 2007.

[6] Nicolas Boullé, Seick Kim, Tianyi Shi, and Alex Townsend. Learning green's functions associated with time-dependent partial differential equations. *Journal of Machine Learning Research*, 23(218):1–34, 2022.

[7] Yadi Cao, Yuxuan Liu, Liu Yang, Rose Yu, Hayden Schaeffer, and Stanley Osher. Vicon: Vision in-context operator networks for multi-physics fluid dynamics prediction. *arXiv preprint arXiv:2411.16063*, 2024.

[8] Javier Castro. The kolmogorov infinite dimensional equation in a hilbert space via deep learning methods. *Journal of Mathematical Analysis and Applications*, 527(2):127413, 2023.

[9] Javier Castro, Claudio Muñoz, and Nicolás Valenzuela. The calderón's problem via deeponets. *Vietnam Journal of Mathematics*, 52(3):775–806, 2024.

[10] Chuanqi Chen and Jinlong Wu. Neural operator for modeling dynamic systems. *arXiv preprint arXiv:2306.XXXX*, 2023.

[11] T. Chen and H. Chen. Approximations of continuous functionals by neural networks with application to dynamic systems. *IEEE Transactions on Neural Networks*, 4(6):910–918, 1993.

[12] 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.

[13] Maarten V. de Hoop, Daniel Zhengyu Huang, Elizabeth Qian, and Andrew M. Stuart. The cost-accuracy trade-off in operator learning with neural networks, 2022.

[14] J. Dugundji. An extension of Tietze's theorem. *Pacific Journal of Mathematics*, 1(3):353 – 367, 1951.

[15] Takashi Furuya, Michael Anthony Puthawala, Matti Lassas, and Maarten V. de Hoop. Globally injective and bijective neural operators. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[16] Craig R. Gin, Daniel E. Shea, Steven L. Brunton, and J. Nathan Kutz. Deepgreen: deep learning of green's functions for nonlinear boundary value problems. *Scientific Reports*, 11(1):21614, 2021.

[17] Somdatta Goswami, Aniruddha Bora, Yue Yu, and George Em Karniadakis. *Physics-Informed Deep Neural Operator Networks*, pages 219–254. Springer International Publishing, Cham, 2023.

[18] Alex Graves, Abdel rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

[19] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[21] Lukas Herrmann, Christoph Schwab, and Jakob Zech. Neural and spectral operator surrogates: unified construction and expression rate bounds. *Advances in Computational Mathematics*, 50(4):72, 2024.

[22] Daniel Zhengyu Huang, Nicholas H. Nelsen, and Margaret Trautner. An operator learning perspective on parameter-to-observable maps. *Foundations of Data Science*, 7(1):163–225, 2025.

[23] Yasamin Jalalian, Juan Felipe Osorio Ramirez, Alexander Hsu, Bamdad Hosseini, and Houman Owhadi. Data-efficient kernel methods for learning differential equations and their solution operators: Algorithms and error analysis, 2025.

[24] Zhongyi Jiang, Min Zhu, Dongzhuo Li, Qiuzi Li, Yanhua O. Yuan, and Lu Lu. Fourier-mionet: Fourier-enhanced multiple-input neural operators for multiphase modeling of geological carbon sequestration. *arXiv preprint arXiv:2303.04778*, 2023.

[25] Pengzhan Jin, Shuai Meng, and Lu Lu. Mionet: Learning multiple-input operators via tensor product. *SIAM Journal on Scientific Computing*, 44(6):A3490–A3514, 2022.

[26] Derek Jollie, Jingmin Sun, Zecheng Zhang, and Hayden Schaeffer. Time-series forecasting and refinement within a multimodal pde foundation model. *Journal of Machine Learning for Modeling and Computing*, 6(2):77–89, 2025.

[27] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[28] Alexander S. Kechris. *Classical Descriptive Set Theory*, volume 156 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1995.

[29] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric pde problems with artificial neural networks. *European Journal of Applied Mathematics*, 32(3):421–435, 2021.

[30] Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for fourier neural operators. *J. Mach. Learn. Res.*, 22(1), January 2021.

[31] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: learning maps between function spaces with applications to pdes. *J. Mach. Learn. Res.*, 24(1), January 2023.

[32] Nikola B. Kovachki, Samuel Lanthaler, and Andrew M. Stuart. Chapter 9 - operator learning: Algorithms and analysis. In Siddhartha Mishra and Alex Townsend, editors, *Numerical Analysis Meets Machine Learning*, volume 25 of *Handbook of Numerical Analysis*, pages 419–467. Elsevier, 2024.

[33] Samuel Lanthaler. Operator learning with pca-net: upper and lower complexity bounds. *J. Mach. Learn. Res.*, 24(1), January 2023.

[34] Samuel Lanthaler, Siddhartha Mishra, and George E Karniadakis. Error estimates for deeponets: a deep learning framework in infinite dimensions. *Transactions of Mathematics and Its Applications*, 6(1):tnac001, 03 2022.

[35] Samuel Lanthaler and Andrew M Stuart. The parametric complexity of operator learning. *IMA Journal of Numerical Analysis*, page draf028, 08 2025.

[36] Jose Antonio Lara Benitez, Takashi Furuya, Florian Faucher, Anastasis Kratsios, Xavier Tricoche, and Maarten V. de Hoop. Out-of-distributional risk bounds for neural operators with applications to the helmholtz equation. *J. Comput. Phys.*, 513(C), September 2024.

[37] Bian Li, Hanchen Wang, Shihang Feng, Xiu Yang, and Youzuo Lin. Solving seismic wave equations on variable velocity models with fourier neural operator. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–18, 2023.

[38] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[39] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021. preprint arXiv:2010.08895.

[40] Hao Liu, Haizhao Yang, Minshuo Chen, Tuo Zhao, and Wenjing Liao. Deep nonparametric estimation of operators between infinite dimensional spaces. *J. Mach. Learn. Res.*, 25(1), January 2024.

[41] Hao Liu, Zecheng Zhang, Wenjing Liao, and Hayden Schaeffer. Neural scaling laws of deep relu and deep operator network: A theoretical study, 2024.

[42] Yuxuan Liu, Jingmin Sun, Xinjie He, Griffin Pinney, Zecheng Zhang, and Hayden Schaeffer. Prose-fd: A multimodal pde foundation model for learning multiple operators for forecasting fluid dynamics. *arXiv preprint arXiv:2409.09811*, 2024.

[43] Yuxuan Liu, Jingmin Sun, and Hayden Schaeffer. Bcat: A block causal transformer for pde foundation models for fluid dynamics. *arXiv preprint arXiv:2501.18972*, 2025.

[44] Yuxuan Liu, Zecheng Zhang, and Hayden Schaeffer. Prose: Predicting multiple operators and symbolic expressions using multimodal transformers. *Neural Networks*, 180:106707, 2024.

[45] 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.

[46] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

[47] Carlo Marcati and Christoph Schwab. Exponential convergence of deep operator networks for elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, 61(3):1513–1545, 2023.

[48] Ivan Markovsky. *Low Rank Approximation: Algorithms, Implementation, Applications*. Communications and Control Engineering. Springer London, 1st edition, 2012.

[49] Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Holden Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Géraud Krawezik, Francois Lanusse, et al. Multiple physics pretraining for physical surrogate models. *arXiv preprint arXiv:2310.02994*, 2023.

[50] István Mező. *The Lambert W Function: Its Generalizations and Applications*. Routledge / Chapman and Hall, London / New York, 2022.

[51] Christian Moya, Guang Lin, Tianqiao Zhao, and Meng Yue. On approximating the dynamic response of synchronous generators via operator learning: A step towards building deep operator-based power grid simulators. *arXiv preprint arXiv:2301.12538*, 2023.

[52] Elisa Negrini, Yuxuan Liu, Liu Yang, Stanley J Osher, and Hayden Schaeffer. A multimodal pde foundation model for prediction and scientific text descriptions. *arXiv preprint arXiv:2502.06026*, 2025.

[53] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, and Anima Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[54] Jingmin Sun, Yuxuan Liu, Zecheng Zhang, and Hayden Schaeffer. Towards a foundation model for partial differential equations: Multioperator learning and extrapolation. *Physical Review E*, 111(3):035304, 2025.

[55] Jingmin Sun, Zecheng Zhang, and Hayden Schaeffer. Lemon: Learning to learn multi-operator networks, 2025.

[56] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: an extensive benchmark for scientific machine learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. doi: 10.5555/3600270.3600387.

[57] Liu Yang, Siting Liu, Tingwei Meng, and Stanley J Osher. In-context operator learning with data prompts for differential equation problems. *Proceedings of the National Academy of Sciences*, 120(39):e2310142120, 2023.

[58] Liu Yang, Tingwei Meng, Siting Liu, and Stanley J Osher. Prompting in-context operator learning with sensor data, equations, and natural language. *arXiv preprint arXiv:2308.05061*, 2023.

[59] Zhanhong Ye, Zining Liu, Bingyang Wu, Hongjie Jiang, Leheng Chen, Minyan Zhang, Xiang Huang, Qinghe Meng Zou, Hongsheng Liu, and Bin Dong. Pdeformer-2: A versatile foundation model for two-dimensional partial differential equations. *arXiv preprint arXiv:2507.15409*, 2025.

[60] Benjamin J Zhang, Siting Liu, Stanley J Osher, and Markos A Katsoulakis. Probabilistic operator learning: generative modeling and uncertainty quantification for foundation models of differential equations. *arXiv preprint arXiv:2509.05186*, 2025.

[61] Zecheng Zhang. Modno: Multi-operator learning with distributed neural operators. *Computer Methods in Applied Mechanics and Engineering*, 431:117229, 2024.

[62] Zecheng Zhang, Wing Tat Leung, and Hayden Schaeffer. A discretization-invariant extension and analysis of some deep operator networks. *Journal of Computational and Applied Mathematics*, 456:116226, 2025.

[63] Zecheng Zhang, Christian Moya, Lu Lu, Guang Lin, and Hayden Schaeffer. D2no: Efficient handling of heterogeneous input function spaces with distributed deep neural operators. *Computer Methods in Applied Mechanics and Engineering*, 428:117084, 2024.

[64] Zecheng Zhang, Leung Wing Tat, and Hayden Schaeffer. Belnet: basis enhanced learning, a mesh-free neural operator. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 479(2276):20230043, 2023.

[65] Min Zhu, Shihang Feng, Youzuo Lin, and Lu Lu. Fourier-deeponet: Fourier-enhanced deep operator networks for full waveform inversion with improved accuracy, generalizability, and robustness. *arXiv preprint arXiv:2305.17289*, 2023.

A Experiment Setup

A.1 Training

The models are trained using the AdamW optimizer for 50 epochs where each epoch is 2K steps. On 2 NVIDIA GeForce RTX 4090 GPUs with 24 GB memory, Table 14 indicates the training time for different models and configurations.

Table 14: Training time for different models and configurations.

MONet	30 min
MNO-S	30 min
MNO-L	1 h 9 min
DeepONet	29 min
DeepONet-C	29 min
MIONet	47 min

A.2 Hyperparameters

The optimizer hyperparameters are summarized in Table 15.

Table 15: Optimizer hyperparameters.

Learning rate	10^{-4}	Gradient norm clip	1.0
Scheduler	Cosine	Weight decay	10^{-4}
Batch data size	150	Warmup steps	10% of total steps
Batch task size	5		