

A discretization-invariant extension and analysis of some deep operator networks

Zecheng Zhang*, Wing Tat Leung †, Hayden Schaeffer ‡

July 20, 2023

Abstract

We present a generalized version of the discretization-invariant neural operator in [43] and prove that the network is a universal approximation in the operator sense. Moreover, by incorporating additional terms in the architecture, we establish a connection between this discretization-invariant neural operator network and those discussed in [7] and [26]. The discretization-invariance property of the operator network implies that different input functions can be sampled using various sensor locations within the same training and testing phases. Additionally, since the network learns a “basis” for the input and output function spaces, our approach enables the evaluation of input functions on different discretizations. To evaluate the performance of the proposed discretization-invariant neural operator, we focus on challenging examples from multiscale partial differential equations. Our experimental results indicate that the method achieves lower prediction errors compared to previous networks and benefits from its discretization-invariant property.

1 Introduction

Operator learning [7, 26, 43, 21] is an approach to approximate mappings between two function spaces, and can be seen as a generalization of the standard machine learning architectures. These approaches have gained significant attention in recent years, particularly for their applicability to scientific computing applications that require approximations of solution operators for spatiotemporal dynamics [26, 43, 21]. Operator networks have been used to approximate solutions to parametric partial differential equations (PDEs) [27, 43, 16, 2]. Furthermore, operator learning has been employed for modeling control problems in dynamical systems [24, 31]. Additionally, operator learning can be used to train models with varying levels of fidelity [14, 28, 13] and for data-driven prediction [33, 5].

Operator learning was initially proposed in [7, 6] using a shallow network architecture and was shown to be a universal approximation for nonlinear continuous operators. Building upon this work, the Deep Operator Neural Network (DON) was developed in [26, 16] and extended the network in [7] to deep architectures. In particular, [26] extend the two layer operator network to networks of arbitrary depth, while [16] generalized the operator network to handle multi-input and multi-output problems. Convergence analysis of DON can be found in [19, 20]. Additionally,

*Department of Mathematics, Florida State University, Tallahassee, FL 32304, USA. (Email: zecheng.zhang.math@gmail.com)

†Department of Mathematics, City University Hong Kong, Hong Kong, China. (Email: wtleung27@cityu.edu.hk)

‡Department of Mathematics, UCLA, Los Angeles, CA 90095. (Email: hayden@math.ucla.edu)

[23, 25] investigated operator learning in the presence of noise and proposed accelerated training methodologies for DON. Another approach, known as the Fourier neural operator (FNO), was introduced and analyzed in [21, 18, 42, 45]. FNO uses the Fourier transformation and inverse Fourier transformation on the kernel integral approximation for approximating an operator. A comparison between DON and FNO in terms of theory and computational accuracy is presented in [27]. Additional noteworthy operator learning frameworks include [43, 32, 34, 45].

The choice of discretizations and domains is crucial in operator learning, as both the input and output are functions. A neural operator is said to be input (output) discretization-invariant if the network can handle varying discretizations for the input (output) function. This means that the input (or output) functions can be evaluated on different grids during both training and testing phases [43, 21, 22]. This property is sometimes referred to as resolution-invariant or a non-uniform mesh approach. A discretization-invariant method is one that does not require the (1) the input discretization to be fixed, (2) the output discretization to be fixed, and (3) the input and output spaces to be the same [43]. The Basis Enhanced Learning operator network (BelNet) was developed as a discretization-invariant approach. BelNet shares similarities with DON [30] as it learns a representation for the output function space through the *construction net* (see Figure 3). However, BelNet also learns the projection of the input functions onto a set of “basis” terms obtained during training using the *projection net*. The architecture of BelNet resembles an encoder-decoder, where the projection net acts as an encoder and the subsequent layers in the network decode the reduced-order model produced by the earlier layers. Numerical experiments presented in [43] demonstrate that BelNet can approximate nonlinear operators without relying on fixed grids, although this property has not been formally proven.

In this work, we present a generalization of BelNet and provide a proof of the universal approximation theorem in the operator sense. We introduce a sub-network structure called the *nonlinear net*, which enhances the flexibility of the architecture. This nonlinear net is motivated by the universal approximation theorem proposed by [7]. Our proof strategy involves establishing connections between various discretizations through a proxy sampling that is fixed, thus allowing us to leverage existing approximation theorems. Furthermore, our proof introduces a more comprehensive approach through the encoder-decoder structure. Specifically, we demonstrate that our model obtains a reduced order model within a subnetwork, which while often stated in the literature, has not been shown. To distinguish our proposed network from previous work, we refer to the BelNet framework introduced in [43] as “vanilla BelNet,” while our new network is referred to as “BelNet.” The proposed network is related to the parallel work of [15], who developed a similar structure based on a universal approximation result for operators with varying sensors. However, the analysis in [15] relies on having access to the continuous inner product layer, which does not imply that the conclusions hold for the discrete (approximated) network.

Data plays a pivotal role in augmenting the learning process for physical systems. Notably, one can gain insights into the underlying principles governing physical phenomena through data-discovery [41, 3, 4, 37, 36, 29, 39, 35, 38, 40]. By employing data-driven approaches, these works have effectively extracted and learned valuable information about the intricate dynamics of the physical systems or governing model. This is one potential of data-driven methods for scientific enhancement and for modeling of complex physical processes. Recently, related methods for solving multiscale problems were proposed, wherein real observation data is employed to enhance a coarse-scale multiscale model [44]. The approach involves training an operator that can map the coarse-scale solution (input function) to a finer-scale solution (output function) using the finer-scale solution obtained at specific locations within the domain.

To evaluate the performance of BelNet, we examine its effectiveness in solving the viscous Burgers’ equation and learning the mapping between two multiscale models. Previous work [44]

demonstrated the concept of mapping between coarse and fine-scale solutions using operator learning, showing its applicability in various examples using DON. In our work, we show that BelNet offers more flexibility in selecting observation points for the coarse-scale input functions. To introduce additional complexity to the problem, we employ random sensors to sample the input functions. The corresponding section provides detailed results.

1.1 Contributions

We summarize the key contributions in this work below.

1. We introduce a generalization of the vanilla BelNet [43], a neural operator that is discretization invariant, and a proof of the universal approximation property for this extended model.
2. The new BelNet extends the universal approximation results of [7, 6, 26].
3. We show a learning approach to map between two multiscale models on different grids and showcase the effectiveness of BelNet in handling challenging observation data.

The rest of the paper is organized as follow. In Section 2, we will review DON, vanilla BelNet, and the extended BelNet. We then present the universal approximation analysis in Section 3. In Section 4, we present numerical experiments.

2 Preliminary Results and Important Lemmata

Let Y be a Banach space and assume that $K_1 \subset Y$ and $K_2 \subset \mathbb{R}$ are both compact. Also, let $V \subset C(K_1)$ be compact and $G : V \rightarrow C(K_2)$ be a continuous and nonlinear operator. DON (see Figure 1) approximates the operator G using a deep neural network. Specifically, in [7] it was shown that for any $\epsilon > 0$, there exists positive integers M, N, K , constants $c_i^k, \zeta_k, \theta_i^k, \varepsilon_{ij}^k \in \mathbb{R}$, points $\omega_k \in \mathbb{R}^d, y_j \in K_1$, where $i \in [M], k \in [K], j \in [N]$ such that

$$\left| G(u)(x) - \sum_{k=1}^K \sum_{i=1}^M c_i^k g \left(\sum_{j=1}^N \varepsilon_{ij}^k u(y_j) + \theta_i^k \right) g(\omega_k \cdot x + \zeta_k) \right| < \epsilon$$

holds for all $u \in V$ and $x \in K_2$.

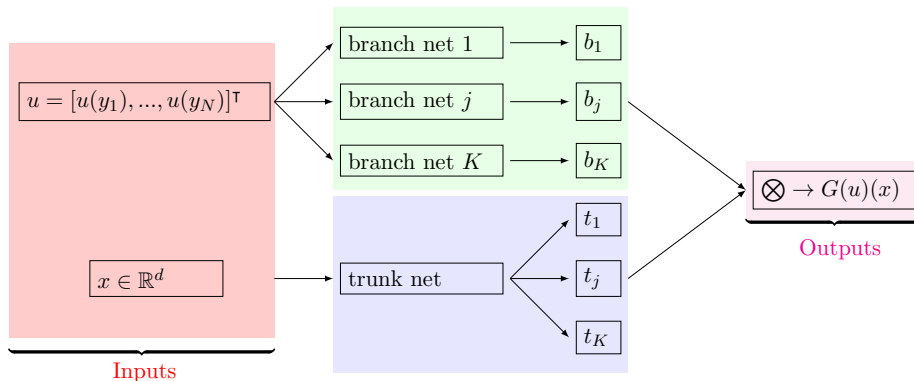


Figure 1: Stacked version DON. \otimes denotes the inner product in \mathbb{R}^K .

The sensors are denoted by $y_i \in K_1$ and the input function u is evaluated on the sensors. That is $y = [y_1, \dots, y_N]^\top$ is the collection of points that represent the discrete grid for the input functions. Theorem 5 in [7] states that the sensors for all input functions u must be the same (i.e. the input discretization must be fixed). This constraint imposes limitations on the applicability of the DON framework in the regime where one does not have control over the input functions or the sensor locations. Ideally, it would be desirable for the operator learning approach to allow for different input functions u to have varying or non-uniform discretization, i.e. to have a discretization-invariant method.

The vanilla BelNet, as displayed in Figure 2, learns the basis functions for both the input and output function spaces. The authors provide an explanation of the network structure by examining a special case (linear) and validating the discretization-invariant property through various numerical experiments. Mathematically, let us introduce weights and biases, $q^k \in \mathbb{R}^d$, $W_y^{1,k} \in \mathbb{R}^{N_1 \times N}$, $W_y^{2,k} \in \mathbb{R}^{N \times N_1}$, $b_x^k \in \mathbb{R}$, and $b_y^k \in \mathbb{R}^{N_1}$, where $k = 1, \dots, K$, and activation functions a_x , a_y and a_u , then the vanilla BelNet, denoted by N_θ , approximates the operator G as follows,

$$G(u)(x) \approx N_\theta(u(y), y)(x) = \sum_{k=1}^K a_x \left((q^k)^\top x + b_x^k \right) a_u \left(\hat{u}^\top W_y^{2,k} \left(a_y (W_y^{1,k} y + b_y^k) \right) \right), \quad (1)$$

for $x \in K_2 \subset \mathbb{R}$, $u \in V$, and where $y = [y_1, \dots, y_N]^\top \subset K_1^N$ and $\hat{u} = [u(y_1), \dots, u(y_N)]^\top$. The network structure is also displayed in Figure 2. We do not assume that the sensors $y_i \in K_1$ are uniform for all input functions (i.e. they are not fixed).

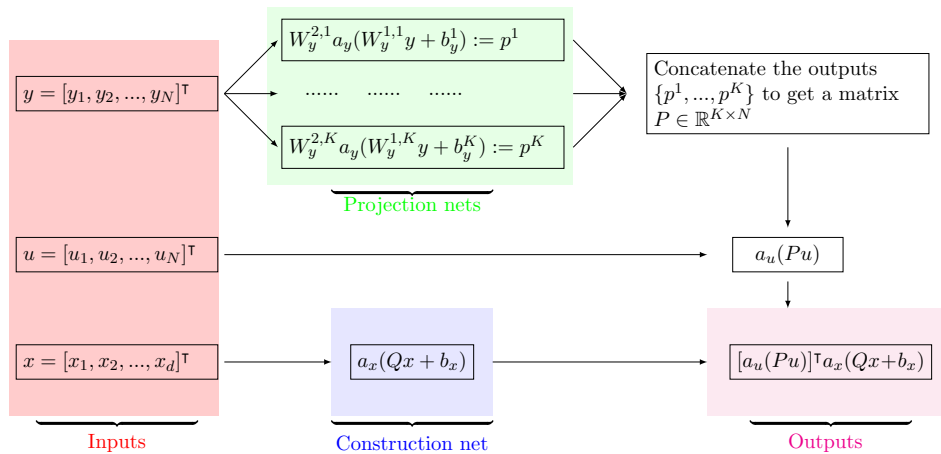


Figure 2: Plot of the vanilla BelNet structure. Projection nets are K independent fully connected neural network with weights and bias $W_y^{2,k} \in \mathbb{R}^{N \times N_1}$, $W_y^{1,k} \in \mathbb{R}^{N_1 \times N}$ and $b_y^k \in \mathbb{R}^{N_1}$. Construction net is a fully connected neural network with weights and bias $Q \in \mathbb{R}^{K \times d}$ and $b_x \in \mathbb{R}^d$. Here $Q = [q^1, q^2, \dots, q^K]$, where $q^i \in \mathbb{R}^d$ are defined in Equation (1). In addition, a_x, a_y, a_u are activation functions.

The motivation behind the vanilla BelNet stems from Mercer's theorem, which involves approximating the linear operator through a kernel integral formulations [43]. However, in order to introduce nonlinearity, the activation function a_u is incorporated. For flexibility and expressiveness, the authors in [43] included an extra trainable layer prior to the activation function, denoted by the term $a_u(WPu)$. Here, W represents a trainable matrix of the appropriate dimension.

In this work, we generalize the vanilla BelNet and prove the universal approximation theorem. Instead of applying an activation function, we design a network to enforce the nonlinearity, see

Figure 3. The additional subnetwork is theoretically consistent with an operator approximation and is partly motivated through the analysis detailed in Section 3.

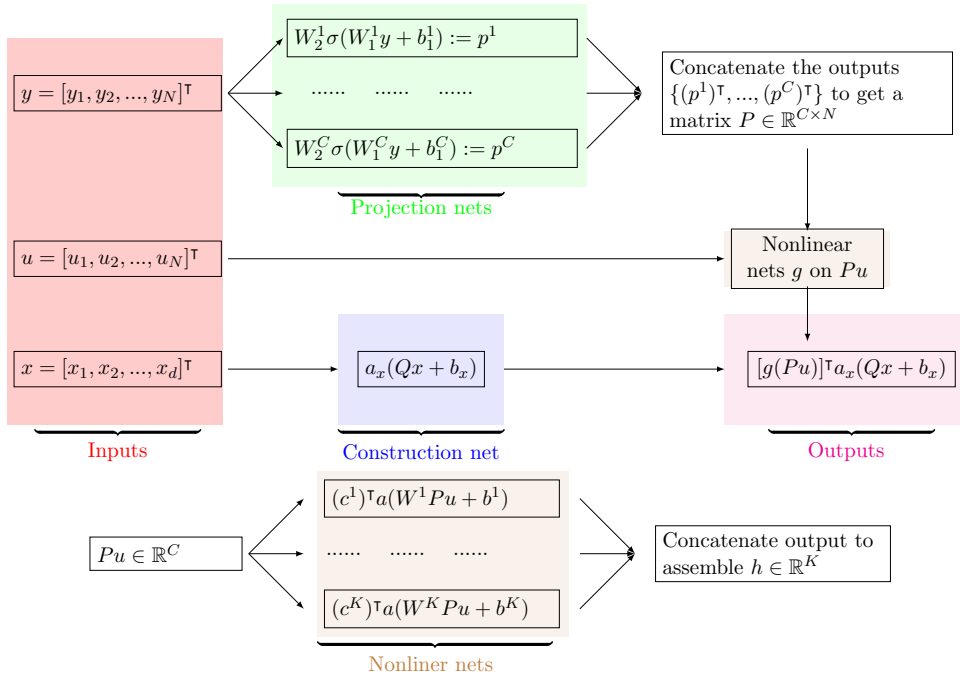


Figure 3: BelNet structure. Projection nets are C independent fully connected neural network with weights and bias $W_1^i \in \mathbb{R}^{N_1 \times N}$, $W_2^i \in \mathbb{R}^{C \times N_1}$ and $b_1^i \in \mathbb{R}^{N_1}$. Construction net is a fully connected neural network with weights and bias $Q \in \mathbb{R}^{K \times d}$ and $b_x \in \mathbb{R}^K$. Here $Q = [q^1, q^2, \dots, q^K]$, where $q^i \in \mathbb{R}^d$. Nonlinear nets are K independent neural networks. Specifically, $c^i \in \mathbb{R}^I$, and $W^i \in \mathbb{R}^{I \times C}$ and $b^i \in \mathbb{R}^I$. In addition, a_x, σ, a are activation functions.

3 Main Results

In this section, we prove the universal approximation theorem of BelNet in the sense of operators.

Definition 3.1. If a function $g : \mathbb{R} \rightarrow \mathbb{R}$ (continuous or discontinuous) satisfies that all linear combinations $\sum_{i=1}^N c_i g(\lambda_i x + \theta_i)$ are dense in $C[a, b]$, where $c_i, \lambda_i, \theta_i \in \mathbb{R}$, then g is called a *Tauber-Wiener (TW) function*.

Theorem 3 from [7] proves the universal approximation for functions. Unlike the universal approximation theorems from [11, 1, 17], the approximation coefficients $c_i(f)$ is a functional which depends on the input function f .

Lemma 3.2 (Theorem 3 from [7]). *Suppose $H \subset \mathbb{R}^d$ is compact, $V \subset C(H)$ is also compact, and $g \in TW$. Let $f \in V$ and for any $\epsilon > 0$, there exists an integer $K > 0$ independent of f , and continuous linear functionals c_i on V such that*

$$\left| f(y) - \sum_{k=1}^K c_k(f) g(w_k \cdot y + b_k) \right| < \epsilon,$$

for all $y \in H$ and $f \in V$.

The following two topological lemmata are used to construct the input function approximation u_k as detailed in Equation 3.

Lemma 3.3 (Lemma 5 from [7]). *Let Y be a Banach space and $H \subset Y$, then H is compact if and only if the following two statements are true:*

1. H is closed.
2. For any $\eta > 0$, there is a η -net $N(\eta) = \{y_1, \dots, y_{m(\delta)}\}$, i.e., for any $y \in H$, there is $y_k \in N(\eta)$ such that $\|y - y_k\| < \eta$.

We recall some properties of compact subsets of continuous functions.

Lemma 3.4 (Lemma 6 from [7]). *$V \subset C(H)$ is compact set in $C(H)$, then it is uniformly bounded and equicontinuous, i.e.,*

1. There is a constant $A > 0$ such that $\|u(y)\|_{C(H)} \leq A$ for all $u \in V$.
2. For all $\epsilon > 0$, there exists $\delta > 0$ such that $|u(y') - u(y'')| < \epsilon$ for all $u \in V$ provided $\|y' - y''\| < \delta$.

Remark 1. *Let f be a continuous functional on V . Pick a sequence $\epsilon_1 > \epsilon_2 > \dots > \epsilon_n \rightarrow 0$, there exists another sequence $\delta_1 > \delta_2 > \dots > \delta_n > 0$, such that, $|f(u) - f(v)| < \epsilon_k$, for all $|u - v| < \delta_k$. By Lemma 3.4, there exists a sequence $\eta_1 > \eta_2 > \dots > \eta_n \rightarrow 0$ such that $|u(y') - u(y'')| < \delta_k$ for all $\|y' - y''\| < \eta_k$ and $u \in V$.*

We can find a sequence $\{z_i\}_{i=1}^{\infty} \subset H$ and a sequence $m(\eta_1) < m(\eta_2) < \dots < m(\eta_n)$ such that the first $m(\eta_k)$ elements $N(\eta_k) = \{z_1, \dots, z_{m(\eta_k)}\}$ is a η_k -net of H . For each η_k -net, and $z_j \in N(\eta_k)$, define a function,

$$T_{k,j}^*(y) = \begin{cases} 1 - \frac{\|y - z_j\|_H}{\eta_k}, & \|y - z_j\|_H \leq \eta_k, \\ 0, & \text{otherwise,} \end{cases}$$

where $y \in H$. Next, we define,

$$T_{k,j}(y) = \frac{T_{k,j}^*(y)}{\sum_{j=1}^{m(\eta_k)} T_{k,j}^*(y)}, \quad (2)$$

and a matrix $T^k \in \mathbb{R}^{m(\eta_k) \times m(\eta_k)}$, where the (i, j) th-entry of T^k is $T_{k,i}(z_j)$. For any $u \in V$, we define a function,

$$u_k(y) = \sum_{j=1}^{m(\eta_k)} u(z_j) T_{k,j}(y), \quad (3)$$

and set $\hat{u}_z^k = [u(z_1), \dots, u(z_k)]^\top$. Furthermore, we define $V_k = \{u_k : u \in V\}$ and $\tilde{V} = V \cup (\cup_{k=1}^{\infty} V_k)$. The next lemma establishes the approximation of u by u_k .

Lemma 3.5 (Lemma 7 from [7]). *For any $u \in V = C(K_1)$, and $\delta_k > 0$, there exists a η_k -net $N(\eta_k) \subset K_1$, and u_k defined as in equation (3) such that,*

$$\|u - u_k\|_{C(K_1)} < \delta_k.$$

The next lemma establishes the universal approximation to continuous functional using a two layer networks.

Lemma 3.6 (Theorem 4 from [7]). *Suppose that $g \in TW$, Y is a Banach space, $K_1 \subset Y$ is compact, and $V \subset C(K_1)$ is also compact. Let f be a continuous functional on V . For any $\epsilon > 0$, there exist integers $I, C > 0$, weight and bias $W \in \mathbb{R}^{I \times C}$ and $c \in \mathbb{R}^I$, $b \in \mathbb{R}^I$ and $\hat{z} = [z_1, \dots, z_C]$ with $z_i \in K_1$ such that,*

$$|f(u) - c^\top g(W\hat{u} + b)| < \epsilon,$$

for all $u \in V$, and $\hat{u}_z = [u(z_1), \dots, u(z_C)]^\top$.

Remark 2. *The proof appears in Theorem 4 from [7], and we discuss an important remark regarding the theorem. The sensors $\{z_i\}_{i=1}^C$ are the evaluation points for the input function space V . They form an η_k -net of K_1 , i.e., $\{z_i\}_{i=1}^C = N(\eta_k) = \{z_1, \dots, z_{m(\eta_k)}\}$, where we denote $C = m(\eta_k)$. The sensors $\{z_i\}_{i=1}^C$, constant C and the η_k -net are determined as follows. For any $\epsilon > 0$, choose $m(\eta_k)$ large enough such that $\|u - u_k\|_{C(Y)} < \delta_k$ implies $|\tilde{f}(u) - \tilde{f}(u_k)| < \epsilon/2$. Here $\tilde{f} \in \tilde{V}$ is the extension of f by the Tietze Extension Theorem, i.e.,*

$$f(w) = \tilde{f}(w), \forall w \in V.$$

One can then define η_k , and η_k -net (the sensors $\{z_i\}_{i=1}^C$) as in Remark 1. We will use the sensors $\{z_i\}_{i=1}^C$ from [7] to establish the universal approximation theorem for BelNet.

To prove the universal approximation theorem of BelNet, the key is to show there is a neural network that can map the function values at arbitrary sensors to \hat{u}_z . In Lemma 3.7, we show a strategy for selecting a set of appropriate sensors and then prove the existence of the neural network.

Lemma 3.7. *Let \hat{z} and C be defined from Lemma 3.6. For any $\epsilon_u > 0$, there exist integers $N, I > 0$, $K_y \subset K_1^N$, and neural networks $\mathcal{N}^i : K_y \rightarrow \mathbb{R}^C$,*

$$\mathcal{N}^i(\hat{y}) = W_2^i a(W_1^i \hat{y} + b_1^i), \quad i \in [N]$$

and $\mathcal{N} : K_y \rightarrow \mathbb{R}^{C \times N}$ defined as $\mathcal{N}(\hat{y}) = [\mathcal{N}^1(\hat{y}), \dots, \mathcal{N}^N(\hat{y})]$, such that

$$\|\hat{u}_z - \mathcal{N}(\hat{y})u(\hat{y})\|_F < \epsilon,$$

where $W_1^i \in \mathbb{R}^{I \times N}$, $W_2^i \in \mathbb{R}^{C \times I}$, and for any $\hat{y} = [y_1, \dots, y_N]^\top \in K_y$.

Proof. For any $\delta > 0$, by Lemma 3.5, there is a sufficiently large integer C_δ such that that $\|u - u_k\|_{C(K_1)} < \delta$. Here $u_k(y) = \sum_{j=1}^{m(\eta_k)} u(r_j)T_{k,j}(y)$ is defined as (3) and $m(\eta_k) = C_\delta$. Moreover we denote $\hat{r} = [r_1, \dots, r_{C_\delta}]^\top$.

For any $N > 0$ and $\hat{y} = [y_1, \dots, y_N]^\top \in K_1^N$ we can define two continuous operators $T_y : K_1^N \rightarrow \mathbb{R}^{N \times C_\delta}$ and $T_z : K_1^C \rightarrow \mathbb{R}^{C \times C_\delta}$ as

$$T_y(\hat{y}) = \begin{pmatrix} T_{k,1}(y_1) & \dots & T_{k,C_\delta}(y_1) \\ \dots & \dots & \dots \\ T_{k,1}(y_N) & \dots & T_{k,C_\delta}(y_N) \end{pmatrix}, \quad T_z(\hat{z}) = \begin{pmatrix} T_{k,1}(z_1) & \dots & T_{k,C_\delta}(z_1) \\ \dots & \dots & \dots \\ T_{k,1}(z_C) & \dots & T_{k,C_\delta}(z_C) \end{pmatrix},$$

where $T_{k,j}$ is defined in Equation (2). For any fixed ϵ_u , δ , and N , we want to construct a subset $K_y \subset K_1^N$, a continuous $v : K_y \rightarrow \mathbb{R}^{C \times N}$, such that,

$$v(\hat{y})T_y(\hat{y}) = T_z(\hat{z}), \quad (4)$$

Let us define $M(\hat{y}) = T_y^\top(\hat{y})T_y(\hat{y})$ and set

$$v(\hat{y}) = T_z(\hat{z})M^{-1}(\hat{y})T_y^\top(\hat{y})$$

for any $\hat{y} \in K_y$. We then define a subset $K_y \subset K_1^N$ as,

$$K_y = \left\{ \hat{y} \in K_1^N, M(\hat{y}) \text{ is invertible and } \|v(\hat{y})\| \leq \frac{\epsilon_u}{2\sqrt{C}\delta^2} - 1 \right\}, \quad (5)$$

where $\|\cdot\|$ is the matrix operator norm. We remark that, for fixed ϵ_u and C , the set K_y is nonempty when $\delta > 0$ is sufficiently small and N is sufficiently large (see Remark 3).

Denote $C_v = \sup_{u \in V} \|u\|_V$, it follows from the universal approximation theorem for functions [11] that, for any $\frac{\epsilon_u}{2\sqrt{NC_v^2}} > 0$, there exist neural networks \mathcal{N}^i of the form $W_2^i \sigma(W_1^i y + b_1^i)$ and $\mathcal{N}(y) = [\mathcal{N}^1(y), \dots, \mathcal{N}^N(y)]$ such that,

$$\|v(y) - \mathcal{N}(y)\|_{C(K_1)} < \frac{\epsilon_u}{2\sqrt{NC_v^2}}. \quad (6)$$

For $\hat{y} = [y_1, \dots, y_N]^\top \in K_y$ and $u_k(y) = \sum_{j=1}^{C_\delta} u(r_j)T_{k,j}(y)$, by multiplying both sides of u_k by $v(\hat{y})$, it follows that,

$$v(\hat{y})u_k(\hat{y}) = \sum_{j=1}^{C_\delta} u(r_j)v(\hat{y})T_{k,j}(\hat{y}) = \sum_{j=1}^{C_\delta} u(r_j)T_{k,j}(\hat{z}_k) = u_k(\hat{z}). \quad (7)$$

By equation (7) and Cauchy-Schwartz, we have the bound:

$$\begin{aligned} \|\mathcal{N}(\hat{y})u(\hat{y}) - u(\hat{z})\|_F &= \|(\mathcal{N}(\hat{y}) - v(\hat{y}) + v(\hat{y}))u(\hat{y}) - u(\hat{z})\|_F \\ &= \|(\mathcal{N}(\hat{y}) - v(\hat{y}))u(\hat{y}) + v(\hat{y})(u(\hat{y}) - u_k(\hat{y}) + u_k(\hat{y})) - u(\hat{z})\|_F \\ &\leq \|(\mathcal{N}(\hat{y}) - v(\hat{y}))u(\hat{y})\|_F + \|v(\hat{y})(u(\hat{y}) - u_k(\hat{y}))\|_F + \|u_k(\hat{z}) - u(\hat{z})\|_F \\ &\leq \|\mathcal{N}(\hat{y}) - v(\hat{y})\| \|u(\hat{y})\|_F + (\|v(\hat{y})\| + 1)\sqrt{C}\delta^2. \end{aligned}$$

Utilizing (5) and (6), the estimation follows. □

Remark 3. We present one example to show K_y is non-empty. Let $\hat{y} = [\hat{r}, \hat{r}, \dots, \hat{r}]$, where we repeat \hat{r} n times, and define $T_y(\hat{y})$ by,

$$T_y(\hat{y}) = \begin{pmatrix} T_r \\ \dots \\ T_r \end{pmatrix}, \text{ where } T_r = T_y(\hat{r}).$$

We have $M = T_y^\top T_y = nT_r^\top T_r$. Thus if T_r has full column rank C_δ , then the matrix M is invertible and it follows that

$$M^{-1}T_y^\top = \frac{1}{n}[(T_r^\top T_r)^{-1}T_r^\top, \dots, (T_r^\top T_r)^{-1}T_r^\top].$$

We estimate the operator norm of $M^{-1}T_y^\top$ by studying its largest singular value σ_1 . We have, $M^{-1}T_y^\top(M^{-1}T_y^\top)^\top = \frac{1}{n}(T_r^\top T_r)^{-1}$ which implies that $\|M^{-1}T_y^\top\| = \sigma_1 \leq \sqrt{\frac{1}{n}\|(T_r^\top T_r)^{-1}\|}$. By letting n be large enough, $\|v\| = \|T_z\| \|M^{-1}T_y^\top\|$ can be sufficiently small, and thus K_y is non-empty.

Remark 4. $M(y) = T_y^\top(y)T_y(y)$, this implies that $\text{rank}(M) = \text{rank}(T_y) \leq \min(C_\delta, N)$. Since $M \in \mathbb{R}^{C_\delta \times C_\delta}$, M is singular if $N < C_\delta$.

Remark 5. \mathcal{N} is the projection net in Figure 3. $\mathcal{N}u$ is the projection coefficients of u onto a set of functions (“basis”) implicitly learned.

Theorem 3.8 (Universal Approximation Theorem for BelNet). Suppose that $a \in TW$, Y is a Banach space, $K_1 \subset Y$, $K_2 \subset \mathbb{R}$ are all compact. $V \subset C(K_1)$ is compact and $G : V \rightarrow C(K_2)$ is continuous and nonlinear. For any $\epsilon > 0$, there exist integers N, C, K, I , weights and biases $W_x^k \in \mathbb{R}^d$, $b_x^k \in \mathbb{R}$, $W^k \in \mathbb{R}^{I \times C}$, $b^k \in \mathbb{R}^I$, $c^k \in \mathbb{R}^I$, subset of sensors $K_y \subset K_1^N$ and a trainable network $\mathcal{N} : K_y \rightarrow \mathbb{R}^{C \times N}$ specified in Lemma 3.7, where K_y satisfies Equation (5). Then the following inequality holds

$$\left| G(u)(x) - \sum_{k=1}^K a(W_x^k \cdot x + b_x^k)(c^k)^\top a(W^k \mathcal{N}(\hat{y})u(\hat{y}) + b^k) \right| < \epsilon,$$

for all $x \in K_2$, $\hat{y} = [y_1, y_2, \dots, y_N]^\top \in K_y$, and $u \in V$.

Proof. Since G is continuous and $V \subset C(K_1)$ is compact, the range $G(V)$ is also compact in $C(K_2)$. By Lemma 3.2, for any $\epsilon > 0$, there exist a positive integer K and linear continuous functional L_k , $W_x^k \in \mathbb{R}^d$, $b_x^k \in \mathbb{R}$ such that

$$\left| G(u)(x) - \sum_{k=1}^K L_k(G(u))a(W_x^k \cdot x + b_x^k) \right| < \frac{\epsilon}{3},$$

for all $x \in K_2$ and $u \in V$. By Lemma 3.6, for all k , there exist integers C, I , $\hat{z} = [z_1, \dots, z_C]^\top$ with $z_i \in K$, $c^k \in \mathbb{R}^I$, $W^k \in \mathbb{R}^{I \times C}$, $b^k \in \mathbb{R}^I$ such that

$$|L_k(G(u)) - (c^k)^\top a(W^k \hat{u}_z + b^k)| < \frac{\epsilon}{3C_u K},$$

where $\hat{u}_z = [u(z_1), \dots, u(z_C)]^\top$ and $C_u = \max_{k, x \in K_2} a(W_x^k \cdot x + b_x^k)$. Therefore, we obtain an approximation to $G(u)(x)$ as in [7] defined by

$$\mathcal{G}(u(\hat{z}))(x) = \sum_{k=1}^K a(W_x^k \cdot x + b_x^k)(c^k)^\top a(W^k u(\hat{z}) + b^k) \quad (8)$$

with $|\mathcal{G}(u(\hat{z}))(x) - G(u)(x)| < \frac{2\epsilon}{3}$. Since a is continuous and K_1^C is compact, we can define uniformly continuous functions $a_k : K_1^C \rightarrow \mathbb{R}$:

$$a_k(\hat{u}) = a(W^k \hat{u} + b^k).$$

Thus, there is an $\epsilon_u > 0$, such that

$$|a_k(\hat{u}') - a_k(\hat{u}'')| < \frac{\epsilon}{3KLC_u} \quad (9)$$

for all $\|\hat{u}' - \hat{u}''\|_F < \epsilon_u$ where $L = \max_k \|c_k\|_{l^1}$.

By Lemma 3.7, there exists N, \mathcal{N} , and $K_y \subset K_1^N$ such that

$$\|u(\hat{z}) - \mathcal{N}(\hat{y})u(\hat{y})\|_F < \epsilon_u \quad \text{for any } y \in K_y. \quad (10)$$

Letting $\hat{y} = [y_1, \dots, y_N] \in K_1^N$, the difference is bounded by

$$\begin{aligned} & \left| G(u)(x) - \sum_{k=1}^K a(W_x^k \cdot x + b_x^k)(c^k)^\top a(W^k \mathcal{N}(\hat{y})u(\hat{y}) + b^k) \right| \\ & \leq \underbrace{|G(u)(x) - \mathcal{G}(u(\hat{z}))(x)|}_{\mathcal{E}_1} \\ & \quad + \underbrace{\left| \mathcal{G}(u(\hat{z}))(x) - \sum_{k=1}^K a(W_x^k \cdot x + b_x^k)(c^k)^\top a(W^k \mathcal{N}(\hat{y})u(\hat{y}) + b^k) \right|}_{\mathcal{E}_2}. \end{aligned}$$

By equations 9, 8 and 10, the second term \mathcal{E}_2 is controlled by:

$$\mathcal{E}_2 = \left| \sum_{k=1}^K a(W_x^k \cdot x + b_x^k)(c^k)^\top (a_k(\hat{z}) - a_k(\mathcal{N}(\hat{y})u(\hat{y}))) \right| < \frac{\epsilon}{3}, \quad (11)$$

and the total approximation follows accordingly. \square

4 Numerical Experiments

We apply our approach to a nonlinear scalar PDE and multiscale PDE problems. Specifically, we first test the proposed BelNet extension on the viscous Burgers' equation. Then we show that BelNet can be used to address some of the difficulties associated with learning multiscale operators. The code and examples will be available when the work is published.

4.1 Parametric Viscous Burgers' Equation

Consider the viscous Burgers' equation with periodic boundary conditions:

$$\begin{aligned} \frac{\partial u_s}{\partial t} + \frac{1}{2} \frac{\partial (u_s^2)}{\partial x} &= \alpha \frac{\partial^2 u_s}{\partial x^2}, \quad x \in [0, 2\pi], t \in [0, 0.3] \\ u_s(x, 0) &= u_s^0(x), \\ u_s(0, t) &= u_s(2\pi, t), \end{aligned}$$

where $u_s^0(x)$ is the initial condition that depends on the parameter s and the viscosity is set to $\alpha = 0.1$. We consider the operator that maps from the initial condition to the terminal solution at $t = 0.3$.

Training Data: In order to obtain more variability between initial samples for the training phase and to include different levels of steepness in the derivative of the initial data, we generate the initial conditions as follows. We first compute a short-time solution ($t = 0.1$) to Burgers' equation using the periodic boundary conditions, set the viscosity to zero, and use the initial condition $s \sin(x)$ where $s \in [0, 4]$. The solution of the system at $t = 0.1$ is then used as the initial condition u_s^0 (resetting time to zero); see the yellow and blue curves in Figure 4 as a display.

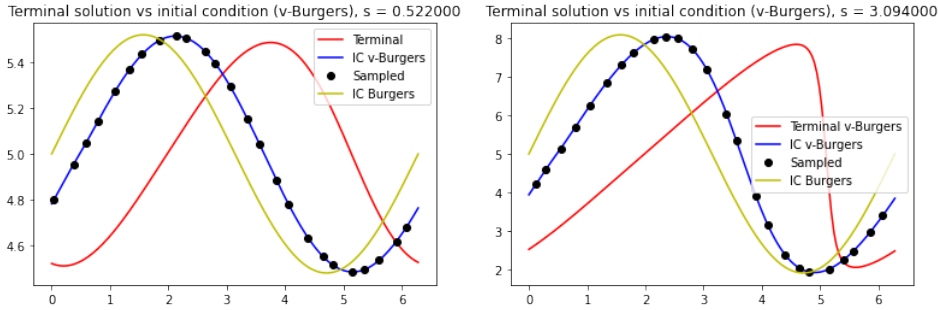


Figure 4: Plots of two solutions to the viscous Burgers’ equation with our initialization procedure. Note that each example’s sampling points (i.e. the sensors represented by the black dots) for the initial condition differ. The yellow curves are used to generate the initial conditions for the model problem (viscous Burgers’ equation). The initial conditions for the viscous Burgers’ equation are displayed in blue.

The mesh for the input data is as follows. Each initial condition (input function) has 25 sensors, and we used a total of 200 initial conditions for training. For each initial condition, the true system is evolved up to time $t = 0.3$, and a total of 5 time-stamps are collected (the terminal time is not included). Therefore the space-time mesh contains 25-by-5 total sample locations for each initial condition.

Testing and observations: We train 100 independent models with the same training dataset, test on the same dataset of 500 samples and compute the average relative error of 100 predictions. To test the neural operators’ ability to forecast future states, we do not include the solution at the terminal time $t = 0.3$ in the training dataset. For testing, we use solutions from 500 initial conditions and test each neural operator on the solution at the terminal time with a finer mesh of 151 grid points. We present the relative errors in Table 1. We compare BelNet with the vanilla BelNet, which was previously shown to be more accurate than comparable models [43]. With fewer trainable parameters, listed in Table 1, BelNet obtains a small prediction error than the vanilla BelNet.

Model	Relative Error	Parameter Count
vanilla BelNet	1.42%	102.93K
BelNet	1.32%	96.5K

Table 1: Relative errors and trainable parameter counts for viscous Burgers equation. The top row is the vanilla BelNet, while the second row is the BelNet. We perform 100 independent experiments and present the average relative errors.

4.2 Multiscale Operator Learning

We test BelNet’s performance on the multiscale operator learning problem. In particular, we apply BelNet to improve a coarse-scale (low-accuracy) solution from a multiscale PDE solver. This problem was introduced in [44] with the DON framework. Let u_0 denote a low-accuracy coarse-scale solution of a given PDE; the target is to construct an operator G such that $G(u_0)(\cdot)$ is a fine-scale solution of the PDE.

To learn the operator, we assume that some observed fine-scale solution data is available. If we denote an approximation to the input function u_0 as \hat{u}_0 and $u(x_i)$ as the fine-scale observed solution at x_i , the dataset for training can then be denoted as $\{x_i, \hat{u}_0, u(x_i)\}_{i=1}^{N_p}$. We can then

construct the loss function as,

$$\sum_{i=1}^{N_p} \|u(x_i) - G_\theta(\hat{u}_0)(x_i)\|^2, \quad (12)$$

where G_θ denotes the neural network with trainable parameter θ .

Since DON is not discretization invariant, i.e., the input function must be discretized in the same way, \hat{u}_0 is then sampled in the same way for all training samples. This limits the potential accuracy as seen in numerical tests. To fix this, the authors of [44] used a localized patch discretization of u_0 which was shown to improve the performance of the DON approximation. However, this is theoretically inconsistent with the DON framework.

Since BelNet is discretization-invariant, a local patch discretize of u_0 is theoretically consistent. Let us denote P_i as a patch (neighborhood) around at x_i , which is the observed solution coordinate. For example, a three-point patch for a 1d problem is $P_i = \{x_i - h_i^1, x_i, x_i + g_i^1\}$, where h_i^1 and g_i^1 are real numbers. The patch is used to discretize the input function u_0 , i.e., $\hat{u}_i = u_0|_{P_i} = [u_0(x_i - h_i^1), u_0(x_i), u_0(x_i + g_i^1)]^\top$ is the local discretization of u_0 . To make the problem more challenging, we assume h_i and g_i are different for all i . We present a 2D display of a patch and the candidate sensors' position in Figure 5.

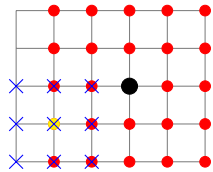


Figure 5: Plot of a 5×5 patch (red dots) centered at an observation point (black dot). To make the problem more challenging, we randomize the sensor position. Specifically, we randomly place a sensor in a neighborhood centered at each red dot. Blue crosses are all candidate locations to place sensors for the yellow dot, we uniformly pick one blue 'x' to place one sensor.

4.2.1 One dimensional elliptic equation

We first study a 1D example for which we can obtain an exact homogenized solution u_0 . In particular, let us consider the following equation:

$$-\frac{d}{dx} \left(\kappa(x/\epsilon) \frac{du}{dx} \right) = f, \quad x \in [0, 1],$$

$$u(0) = u(1) = 0,$$

where $\kappa(x) = 0.5 \sin(2\pi \frac{x}{\epsilon}) + 0.8$ and $f(x) = 0.5$. We plot the multiscale permeability $\kappa(x)$ and the reference solution in Figure (6).

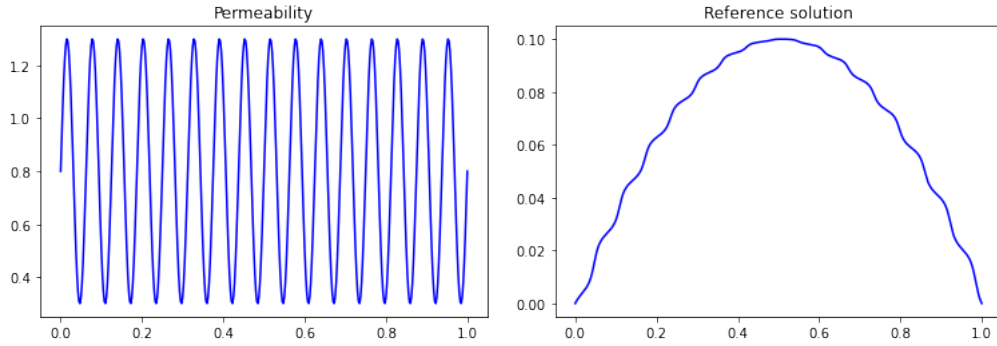


Figure 6: 1D elliptic. Left: permeability κ . Right: reference solution.

The relative error of the homogenized solution is 0.07%; we use exact solutions $u(x_i)$ as the observations, x_i are uniformly distributed, and $N_p = 16$ points are used in total. We use the oversampling trick which employs a patch of coarse-scale solutions to capture the input function (see Figure 5), and the result is presented in Figure 7.

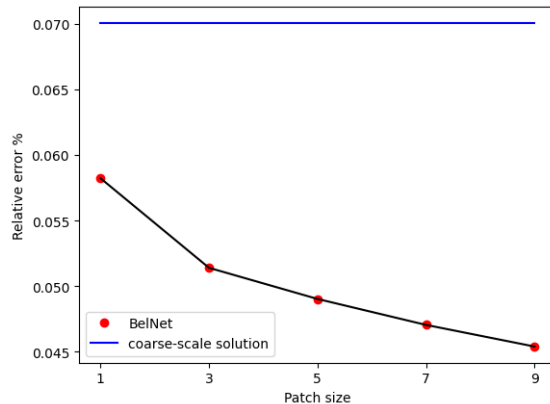


Figure 7: Relative errors with respect to different patch sizes. For each patch size, we perform 100 independent experiments and present the average relative error. All input functions (low accuracy solution) of each experiment do not share the discretization, and the discretizations for the same input function are not the same in 100 independent experiments. The larger the patch size, the more flexibility in sampling the input function, and hence is more challenging to train. We do not observe an increase of the relative error with respect to the patch, which also implies that BelNet is discretization-invariant for this example.

Settings, Observations, and Comments: We use $N_p = 16$ exact solutions uniformly distributed in the domain to improve u_0 . It is important to note that our focus does not involve studying the decay of error in relation to the number of observation points, and for a comprehensive investigation, please refer to the work by [44].

As the size of the patch expands, there is increased flexibility in terms of sensor placement for sampling the input function. Specifically, we consider the placement of 1, 3, 5, 7, and 9 sensors, respectively. The locations of the sensors are randomized for each local patch, denoted as h_i^j and g_i^j , which vary across different samples of $u_0(x_i)$. Refer to Figure 5 for a display of this randomization process. With an increasing number of sensors, the various possible discretizations of each sample u_0 become more numerous, resulting in a more challenging training scenario when dealing with larger patches. However, the results presented in Figure 7 demonstrate that BelNet

exhibits discretization invariance, as the accuracy remains unaffected and improves with patch size.

4.2.2 2D elliptic equation with one fast variable

We consider the following 2D elliptic equation:

$$-\nabla \cdot (\kappa(x/\epsilon)\nabla u) = f, x \in \Omega = [0, 1]^2, \tag{13}$$

$$u(x) = 0, x \in \partial\Omega, \tag{14}$$

where $\kappa(x/\epsilon) = 2 + \sin(2\pi x/\epsilon) \cos(2\pi y/\epsilon)$ and $\epsilon = \frac{1}{8}$. We display $\kappa(x)$ in Figure (8).

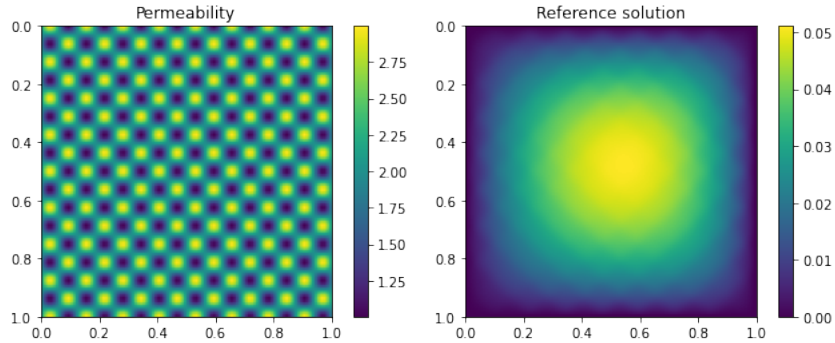


Figure 8: 2D elliptic with 1 fast variable. Left: permeability κ . Right: reference solution.

We measure the input function (the low-accuracy solution) by sampling it in a neighborhood around the observation sensor. In order to obtain the low-accuracy solution, we employ mesh-dependent solvers that define all solutions on grid points. As the neighborhood around the grid point sensor expands, there is an increase in the degrees-of-freedom available for sampling the input function. It is important to highlight that the utilization of different discretizations for the input functions poses difficulties during training. Therefore, we require a discretization-invariant tool such as BelNet to address this issue.

Settings and comments on the results: We increase the patch size, perform 100 independent experiments for each patch size and compute the average relative error. In each experiment, the input function value u_0 , representing the low-accuracy solution, was measured by sampling random points within the patch (neighborhood) surrounding the observation point x_i .

The results are displayed in Figure 9. When the patch size is 1, only a single point is used to sample u_0 , rendering it insufficient for an accurate approximation. As the patch size increases, training becomes more challenging due to the increased freedom in sensor placement for sampling u_0 . However, the approximation error decreases (in trend) indicating that BelNet can effectively handle problems with varying input function meshes.

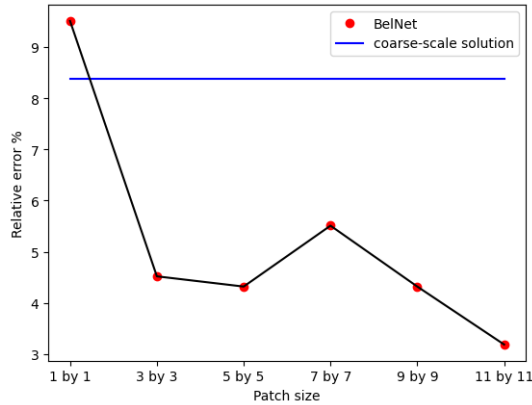


Figure 9: Relative errors with respect to different patch sizes. For each patch size, we conducted 100 independent experiments and calculated the average relative error. It is important to note that the input functions (representing low-accuracy solutions) used in the experiments do not share the same discretization. Furthermore, the discretization of each input function varies across the 100 independent experiments.

4.2.3 2D elliptic multiscale PDE

We consider the same equation (14) but with different permeability κ :

$$\kappa(x, y) = 1 + \frac{\sin(2\pi \frac{x}{\epsilon_0}) \cos(2\pi \frac{y}{\epsilon_1})}{2 + \cos(2\pi \frac{x}{\epsilon_2}) \sin(2\pi \frac{y}{\epsilon_3})} + \frac{\sin(2\pi \frac{x}{\epsilon_4}) \cos(2\pi \frac{y}{\epsilon_5})}{2 + \cos(2\pi \frac{x}{\epsilon_6}) \sin(2\pi \frac{y}{\epsilon_7})},$$

where $\epsilon_0 = \frac{1}{5}$, $\epsilon_1 = \frac{1}{4}$, $\epsilon_2 = \frac{1}{25}$, $\epsilon_3 = \frac{1}{16}$, $\epsilon_4 = \frac{1}{16}$, $\epsilon_5 = \frac{1}{32}$, $\epsilon_6 = \frac{1}{3}$, and $\epsilon_7 = \frac{1}{9}$. We plot the permeability and the solution in Figure (10).

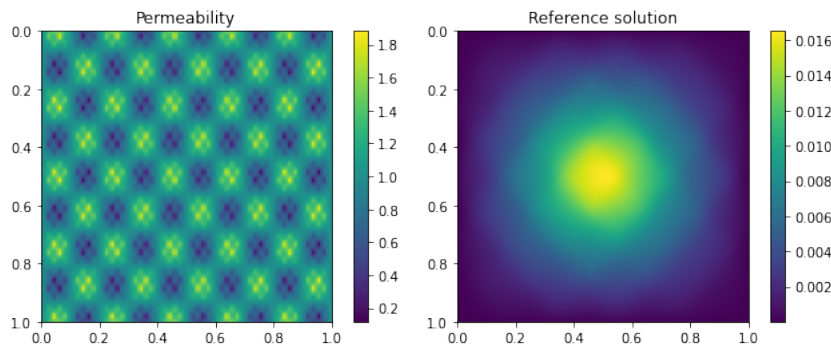


Figure 10: 2D elliptic multiscale PDE. Left: permeability κ . Right: reference solution.

We obtain the coarse-scale solution by the multiscale finite element method with one local basis [12, 9, 10, 8]. We conduct six sets of experiments with patch sizes 1×1 , 3×3 , and 5×5 , 7×7 , 9×9 and 11×11 . We train 100 models for each set of experiments and compute the average relative errors of the last 100 epochs of 100 models. The results are shown in Figure 11.

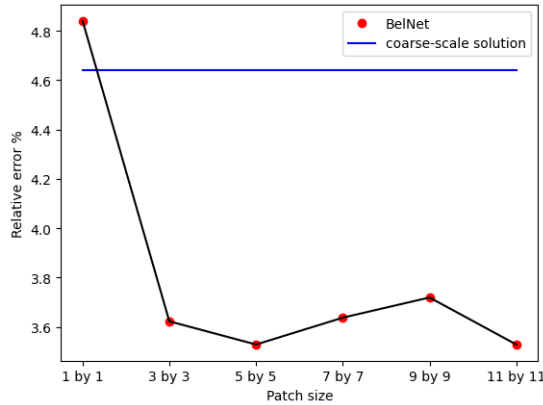


Figure 11: Relative errors with respect to different patch sizes. For each patch size, we perform 100 independent experiments and present the average relative error. It should be noted that the input functions (representing the low accuracy solution) used in the experiments do not share the same discretization. Additionally, the discretization of each input function varies among the 100 independent experiments. Even as the patch size expands, BelNet maintains stability.

Settings, Observations, and Comments: We used a set of 16 input function samples, denoted as $u(x_i)$, which are uniformly distributed to improve the initial function u_0 . When the patch size is 1, all input functions are sampled as $u_0(x_i)$. However, this sampling strategy does not yield a satisfactory approximation to u_0 , resulting in (slightly more) inaccurate prediction as compared to a coarse-scale solution. As the patch size increases from 1×1 to 11×11 , we incorporate a larger number of sensors. Specifically, the number of sensors used is 1, 9, 25, 49, 81, and 121, respectively. Despite the numerical challenge of using more sensors which are non-overlapping, the prediction accuracy does not deteriorate significantly. The error has a slight increase between 5×5 and 9×9 patches which may indicate a saturation of the error within this patch size window (since the error remain around 3.7%). The relative error decreases again as the patch size increases.

5 Conclusion

We generalize the vanilla BelNet architecture proposed in [43] by adding a trainable nonlinear layer in the network. We prove the universal approximation theorem of BelNet in the sense of operators, extending the results of [6, 7, 26]. In particular, we show that BelNet can be viewed as a discretization-invariant extension of the operator networks in [7, 7] which allows for several new applications, particularly to multiscale PDE. In particular, the discretization-invariance property allows for the input functions to be observed at different sensor locations which is often the case for applications where the sensor locations move in time or where fluctuations in data acquisition occurs. For multiscale problems, the randomization of patch location and neighboring points necessitates the use of discretization-invariant learning. We test the performance on high-contrast and multiscale parametric PDE. Our experiments show that BelNet typically obtains about $1.2\times$ to $2\times$ improvement in relative error over the course-scale solution without needing to fully resolve the multiscale problem. Lastly, it is worth noting that part of the theoretical analysis shows that the network obtains a (trained) reduced order model and projection. This is a useful result for assessing the contributions of individual subnetworks within the full operator learning architecture, i.e. peering into the black-box of deep networks for PDE.

6 Acknowledgement

Z. Zhang was supported in part by AFOSR MURI FA9550-21-1-0084. H. Schaeffer was supported in part by AFOSR MURI FA9550-21-1-0084 and an NSF CAREER Award DMS-2331100.

References

- [1] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.
- [2] K. Bhattacharya, B. Hosseini, N. B. Kovachki, and A. M. Stuart. Model reduction and neural networks for parametric pdes. *The SMAI journal of computational mathematics*, 7:121–157, 2021.
- [3] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.
- [4] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Sparse identification of nonlinear dynamics with control (sindyc). *IFAC-PapersOnLine*, 49(18):710–715, 2016.
- [5] Q. Cao, S. Goswami, G. E. Karniadakis, and S. Chakraborty. Deep neural operators can predict the real-time response of floating offshore structures under irregular waves. *arXiv preprint arXiv:2302.06667*, 2023.
- [6] 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.
- [7] T. Chen and H. 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.
- [8] B. Chetverushkin, E. Chung, Y. Efendiev, S.-M. Pun, and Z. Zhang. Computational multiscale methods for quasi-gas dynamic equations. *Journal of Computational Physics*, 440:110352, 2021.
- [9] E. Chung, Y. Efendiev, and T. Y. Hou. Adaptive multiscale model reduction with generalized multiscale finite element methods. *Journal of Computational Physics*, 320:69–95, 2016.
- [10] E. T. Chung, Y. Efendiev, and W. T. Leung. Constraint energy minimizing generalized multiscale finite element method. *Computer Methods in Applied Mechanics and Engineering*, 339:298–319, 2018.
- [11] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.
- [12] Y. Efendiev, J. Galvis, and T. Y. Hou. Generalized multiscale finite element methods (gmsfem). *Journal of computational physics*, 251:116–135, 2013.
- [13] A. Howard, Y. Fu, and P. Stinis. A multifidelity approach to continual learning for physical systems. *arXiv preprint arXiv:2304.03894*, 2023.

- [14] A. A. Howard, M. Perego, G. E. Karniadakis, and P. Stinis. Multifidelity deep operator networks. *arXiv preprint arXiv:2204.09157*, 2022.
- [15] N. Hua and W. Lu. Basis operator network: A neural network-based model for learning nonlinear operators via neural basis. *Neural Networks*, 164:21–37, 2023.
- [16] P. Jin, S. Meng, and L. Lu. Mionet: Learning multiple-input operators via tensor product. *SIAM Journal on Scientific Computing*, 44(6):A3490–A3514, 2022.
- [17] L. K. Jones. A simple lemma on greedy approximation in hilbert space and convergence rates for projection pursuit regression and neural network training. *The annals of Statistics*, pages 608–613, 1992.
- [18] N. Kovachki, S. Lanthaler, and S. Mishra. On universal approximation and error bounds for fourier neural operators. *Journal of Machine Learning Research*, 22:Art–No, 2021.
- [19] S. Lanthaler, S. Mishra, and G. E. Karniadakis. Error estimates for deeponets: A deep learning framework in infinite dimensions. *Transactions of Mathematics and Its Applications*, 6(1):tnac001, 2022.
- [20] S. Lanthaler and A. M. Stuart. The curse of dimensionality in operator learning. *arXiv preprint arXiv:2306.15924*, 2023.
- [21] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.
- [22] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.
- [23] G. Lin, C. Moya, and Z. Zhang. Accelerated replica exchange stochastic gradient langevin diffusion enhanced bayesian deeponet for solving noisy parametric pdes. *arXiv preprint arXiv:2111.02484*, 2021.
- [24] G. Lin, C. Moya, and Z. Zhang. On learning the dynamical response of nonlinear control systems with deep operator networks. *arXiv preprint arXiv:2206.06536*, 2022.
- [25] G. Lin, C. Moya, and Z. Zhang. B-deeponet: An enhanced bayesian deeponet for solving noisy parametric pdes using accelerated replica exchange sgld. *Journal of Computational Physics*, 473:111713, 2023.
- [26] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.
- [27] L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, and G. E. 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.
- [28] L. Lu, R. Pestourie, S. G. Johnson, and G. Romano. Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport. *Physical Review Research*, 4(2):023210, 2022.

- [29] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Inferring biological networks by sparse identification of nonlinear dynamics. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 2(1):52–63, 2016.
- [30] B. Meuris, S. Qadeer, and P. Stinis. Machine-learning-based spectral methods for partial differential equations. *Scientific Reports*, 13(1):1739, 2023.
- [31] K. Michałowska, S. Goswami, G. E. Karniadakis, and S. Riemer-Sørensen. Neural operator learning for long-time integration in dynamical systems with recurrent neural networks. *arXiv preprint arXiv:2303.02243*, 2023.
- [32] T. O’Leary-Roseberry, P. Chen, U. Villa, and O. Ghattas. Derivative informed neural operator: An efficient framework for high-dimensional parametric derivative learning. *arXiv preprint arXiv:2206.10745*, 2022.
- [33] J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.
- [34] E. Qian, I.-G. Farcas, and K. Willcox. Reduced operator inference for nonlinear partial differential equations. *SIAM Journal on Scientific Computing*, 44(4):A1934–A1959, 2022.
- [35] M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.
- [36] H. Schaeffer. Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2197):20160446, 2017.
- [37] H. Schaeffer, R. Caffisch, C. D. Hauck, and S. Osher. Sparse dynamics for partial differential equations. *Proceedings of the National Academy of Sciences*, 110(17):6634–6639, 2013.
- [38] H. Schaeffer and S. G. McCalla. Sparse model selection via integral terms. *Physical Review E*, 96(2):023302, 2017.
- [39] H. Schaeffer, G. Tran, and R. Ward. Extracting sparse high-dimensional dynamics from limited data. *SIAM Journal on Applied Mathematics*, 78(6):3279–3295, 2018.
- [40] H. Schaeffer, G. Tran, R. Ward, and L. Zhang. Extracting structured dynamical systems using sparse optimization with very few samples. *Multiscale Modeling & Simulation*, 18(4):1435–1461, 2020.
- [41] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.
- [42] G. Wen, Z. Li, K. Azizzadenesheli, A. Anandkumar, and S. M. Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.
- [43] Z. Zhang, W. T. Leung, and H. Schaeffer. Belnet: Basis enhanced learning, a mesh-free neural operator. *arXiv preprint arXiv:2212.07336*, 2022.
- [44] Z. Zhang, C. Moya, W. T. Leung, G. Lin, and H. Schaeffer. Bayesian deep operator learning for homogenized1 to fine-scale maps for multiscale pde. 2023.

- [45] M. Zhu, S. Feng, Y. Lin, and L. Lu. Fourier-deeponet: Fourier-enhanced deep operator networks for full waveform inversion with improved accuracy, generalizability, and robustness. *arXiv preprint arXiv:2305.17289*, 2023.