PSEUDO-DIFFERENTIAL INTEGRAL OPERATOR FOR LEARNING SOLUTION OPERATORS OF PARTIAL DIFFERENTIAL EQUATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning mapping between two function spaces has attracted considerable research attention. However, learning the solution operator of partial differential equations (PDEs) remains a challenge in scientific computing. Fourier neural operator (FNO) is recently proposed to learn the solution operators with an excellent performance. In this study, we propose a novel *pseudo-differential integral operator* (PDIO) to analyze and generalize the Fourier integral operator in FNO. PDIO is inspired by a pseudo-differential operator, which is a generalization of a differential operator and characterized by a certain symbol. We parameterize the symbol by using a neural network and show that the neural-network-based symbol is contained in a smooth symbol class. Subsequently, we prove that the PDIO is a bounded linear operator, and thus is continuous in the Sobolev space. We combine the PDIO with the neural operator to develop a *pseudo-differential neural operator* (PDNO) to learn the nonlinear solution operator of PDEs. We experimentally validate the effectiveness of the proposed model by using Darcy flow and the Navier-Stokes equation. The results reveal that the proposed PDNO outperforms the existing neural operator approaches in most experiments.

1 INTRODUCTION

In science and engineering, many physical systems and natural phenomena are described by partial differential equations (PDEs) (Courant & Hilbert, 1953). Approximating the solution of the underlying PDEs is critical to understand and predict a system. Conventional numerical methods, such as finite difference methods (FDMs) and finite element methods, involve a trade-off between accuracy and the time required. In many complex systems, it may be highly time-consuming to use numerical methods to obtain accurate solutions. Furthermore, in some cases, the underlying PDE may be unknown.

With remarkable advancements in deep learning, studies have focused on using deep learning to solve PDEs (Nabian & Meidani, 2018; E & Yu, 2018; Sirignano & Spiliopoulos, 2018; Raissi et al., 2019; Hwang et al., 2020; Lee et al., 2021). An example is an operator learning (Guo et al., 2016; Bhatnagar et al., 2019; Khoo et al., 2021), which utilizes neural networks to parameterize the mapping from the parameters (external force, initial, and boundary condition) of the given PDE to the solutions of that PDE. Many studies employed different convolutional neural networks as surrogate models to solve various problems, such as the uncertainty quantification tasks for PDEs (Zhu & Zabaras, 2018; Zhu et al., 2019) and PDE-constrained control problems (Holl et al., 2020; Hwang et al., 2021). Based on the universal approximation theorem of operator (Chen & Chen, 1995), *DeepONet* was introduced by Lu et al. (2019). In follow-up works, an extension model of the DeepONet was proposed in Wang et al. (2021); Kissas et al. (2022).

Another approach to operator learning is a *neural operator*, proposed in Li et al. (2020c;b;a). Li et al. (2020c) proposed an iterative architecture inspired by Green's function of elliptic PDEs. The iterative architecture consists of a linear transformation, an integral operator, and a nonlinear activation function, allowing the architecture to approximate complex nonlinear mapping. An extension of this work, Li et al. (2020b) used a multi-pole method to develop a multi-scale graph structure. Gupta et al. (2021) approximated the kernel of the integral operator using the multiwavelet transform.

Recently, Li et al. (2020a) proposed a *Fourier integral operator* using fast Fourier transform (FFT) to reduce the cost of approximating the integral operator. They directly parametrize the kernel in Fourier integral operator by its Fourier space coefficients which only depends on frequency mode. In this study, we analyze the Fourier integral operator from the perspective of *pseudo-differential operator* (PDO). PDOs are a generalization of linear partial differential operators and have been extensively studied mathematically (Boutet de Monvel, 1971; Hörmander, 2007; Ruzhansky & Turunen, 2009; Taylor, 2017). A *pseudo-differential integral operator* (PDIO) is proposed to generalize the Fourier integral operator in the FNO based on the PDO. A neural network called a symbol network is used to approximate the PDO symbols. The proposed symbol network is contained in a toroidal class of symbols; thus, a PDIO is a continuous operator in the Sobolev space. Furthermore, the PDIO can be applied to the solution operator of time-dependent PDEs using a time-dependent symbol network.

The main contributions of the study are as follows.

- The Fourier integral operator proposed in Li et al. (2020a) is interpreted from a PDO perspective. The symbol of the Fourier integral operator is only depend on ξ and not depend on position x. Furthermore, the symbol may not be contained in a toroidal symbol class so that the Fourier integral operator cannot be guaranteed to be a continuous operator.
- A novel PDIO is proposed based on the PDO to generalize the Fourier integral operator. PDIO approximates the PDO using symbol networks. We show that the proposed symbol network is contained in a toroidal symbol class of PDOs, implying that the PDIO with the symbol network is a continuous operator in the Sobolev space.
- Time-dependent PDIO, a PDIO with time-dependent symbol networks, can be used to approximate the solution operator of time-dependent PDEs. It approximates the solution operator, including the solution for time t, which is not in the training data. Furthermore, it is a continuous-in-time operator.
- A *pseudo-differential neural operator* (PDNO), which consists of a linear combination of our PDIOs combined with the neural operator proposed in Li et al. (2020c), is developed. PDNO outperforms the existing operator learning models, such as the Fourier neural operator in Li et al. (2020a) and the multiwavelet-based operator in Gupta et al. (2021) in hard problems (Darcy flow and Navier-Stokes equation). In particular, the PDNO reduces overfitting compared to other models (Figure 1.

2 Analysis of Fourier integral operator based on pseudo-differential operator

In this study, we aim to approximate an operator $\mathcal{G} : \mathcal{A} \to \mathcal{U}$ between two function spaces $\mathcal{A}$ and $\mathcal{U}$. The operator $\mathcal{G}$ can be considered as the solution operators of various parametric PDEs (See Section 5 for examples). To find a map from the function $f(x) \in \mathcal{A}$ to the solution $u(x) \in \mathcal{U}$, we introduce a neural operator architecture to learn infinite-dimensional operators effectively.

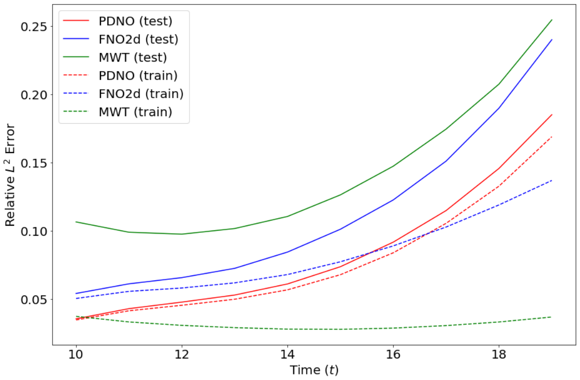

Figure 1: Comparison of the train and the test relative L^2 error by time horizon $t = 10, ..., 19$ on the Navier-Stokes equation with $\nu = 1e-5$. FNO and MWT are highly overfitted, while PDNO is not.

2.1 Neural operator

Inspired by Green's functions of elliptic PDEs, Li et al. (2020c) proposed an iterative *neural operator* to approximate the solution operators of parametric PDEs. First, the input $f(x)$ is lifted to a higher representation $f_0(x) = P(f(x))$. Next, the iterations $f_0 \mapsto f_1 .... \mapsto f_T$ are applied using the update $f_t \mapsto f_{t+1}$ formulated using the following expression:

$$f_{t+1}(x) = \sigma \left(W f_t(x) + \mathcal{K}_\phi[f_t](x) \right), \tag{1}$$

for $t = 0, ..., T-1$, where W is a local linear transformation and σ is a nonlinear activation function. $\mathcal{K}_\phi$ is an integral operator $\mathcal{K}$ parameterized by ϕ as

$$\mathcal{K}_\phi[f_t](x) = \int_D \kappa_\phi(x, y) f_t(y) dy, \tag{2}$$

where D is a bounded domain of input function. The output $u(x) = Q(f_T(x))$ is the projection of $f_T(x)$ by the local transformation Q. There are many studies considering how to choose the kernel function κ_ϕ and how to compute the corresponding integral operator. The integral operator $\mathcal{K}_\phi$ can be parameterized using the message passing on graph networks (Li et al., 2020c). In this work, we focus on the *Fourier integral operator* proposed by Li et al. (2020a).

2.2 Fourier integral operator

Li et al. (2020a) proposed a neural operator structure with the integral operator $\mathcal{K}_R$ called Fourier integral operator. By letting $\kappa(x, y) = \kappa(x - y)$ and using the convolution theroem, they define the Fourier integral operator as follows:

$$\mathcal{K}_R[f_t](x) = \mathcal{F}^{-1}\left[\mathcal{F}[\kappa] \cdot \mathcal{F}[f_t](\xi)\right](x) = \mathcal{F}^{-1}\left[R(\xi) \cdot \mathcal{F}[f_t](\xi)\right](x), \tag{3}$$

where $\mathcal{F}$ is the Fourier transform and $\mathcal{F}^{-1}$ is its inverse. Note that the parameter $R(\xi)$ is directly parameterized on the discrete space $\xi \in \mathbb{Z}^n$. The Fourier integral operator can be interpreted as the general conecept of differential operator, called pseudo-differential operator.

2.3 Pseudo-differential operator

From 1960s, PDOs were derived from differential operators and have been studied. We consider a PDE $\mathcal{L}_a[u(x)] = f(x)$ with a linear differential operator $\mathcal{L}_a = \sum_\alpha c_\alpha D^\alpha$. To find a map T from f to u, we apply the Fourier transform to obtain the following:

$$a(\xi)\hat{u} \overset{\text{def}}{=} \left(\sum_\alpha c_\alpha (i\xi)^\alpha\right) \hat{u} = \hat{f}, \tag{4}$$

where $\xi \in \mathbb{R}^n$ represents variables in the Fourier space and the $\hat{f}(\xi)$ is a Fourier transform of function $f(x)$. If $a(\xi)$ never attains zero, we obtain the solution operator of the PDE as follows:

$$u(x) = T(f)(x) \overset{\text{def}}{=} \int_{\mathbb{R}^n} \frac{1}{a(\xi)} \hat{f}(\xi) e^{2\pi i \xi x} d\xi. \tag{5}$$

A PDO can be defined as a generalization of differential operators by replacing $\frac{1}{a(\xi)}$ with $a(x, \xi)$, called a symbol (Hörmander, 2003; 2007). First, we define a symbol $a(x, \xi)$ and a class of the symbols.

Definition 2.1. Let $0 < \rho \leq 1$ and $0 \leq \delta < 1$. A function $a(x, \xi)$ is called a *Euclidean symbol* on $\mathbb{T}^n \times \mathbb{R}^n$ in a class $S^m_{\rho,\delta}(\mathbb{T}^n \times \mathbb{R}^n)$ if $a(x, \xi)$ is smooth on $\mathbb{T}^n \times \mathbb{R}^n$ and satisfies the following inequality:

$$|\partial_x^\beta \partial_\xi^\alpha a(x, \xi)| \leq c_{\alpha\beta} \langle \xi \rangle^{m - \rho|\alpha| + \delta|\beta|}, \tag{6}$$

for all $\alpha, \beta \in \mathbb{N}_0^n$, and for all $x \in \mathbb{T}^n$ and $\xi \in \mathbb{R}^n$, where a constant $c_{\alpha\beta}$ may depend on α and β but not on x and ξ. Here, $\langle \xi \rangle \overset{\text{def}}{=} (1 + \|\xi\|^2)^{1/2}$ with the Euclidean norm $\|\xi\|$.

The PDO corresponding to the symbol class $S^m_{\rho,\delta}(\mathbb{T}^n \times \mathbb{R}^n)$ can be defined as follows:

Definition 2.2. The *Euclidean PDO* $T_a : \mathcal{A} \to \mathcal{U}$ with the Euclidean symbol $a(x, \xi) \in S^m_{\rho,\delta}(\mathbb{T}^n \times \mathbb{R}^n)$ is defined as follows:

$$T_a(f)(x) = \int_{\mathbb{R}^n} a(x, \xi) \hat{f}(\xi) e^{2\pi i \xi x} d\xi, \tag{7}$$

where $\hat{f}(\xi)$ is the Fourier transform of function $f(x)$.

The Euclidean PDO can be re-written using the Fourier transform as follows:

$$T_a(f)(x) = \mathcal{F}^{-1}\left[a(x, \xi)\mathcal{F}[f](\xi)\right]. \tag{8}$$

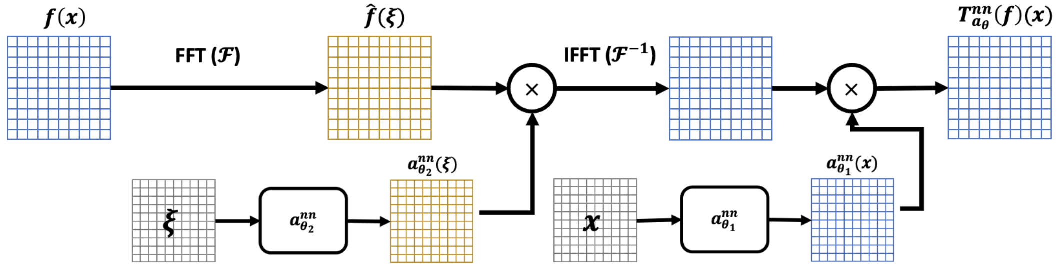

Figure 2: An architecture of a PDIO with symbol networks $a^{nn}_{\theta_1}(x)$ and $a^{nn}_{\theta_2}(\xi)$. Considering that FFT and inverse FFT are used, both the input and output are in the form of uniform mesh. Each value $a^{nn}_{\theta_1}(x)$ and $a^{nn}_{\theta_2}(\xi)$ is obtained from separate neural networks.

2.4 Fourier integral operator and Pseudo-differential operator

Comparing the Fourier integral operator $\mathcal{K}_R$ (equation 3) and the Euclidean PDO T_a (equation 8), there are two main differences. The parameters $R(\xi)$ may not satisfy the condition of the toroidal symbol (equation 10) in Definition 3.1 so that the Fourier integral operator cannot be guaranteed to be a continuous operator. Furthermore, parameters $R(\xi)$ only consider the dependency on ξ, while the symbol $a(x,\xi)$ has a dependency on x. To generalize the Fourier integral operator based on PDO, the key idea of our method is to parametrize the Euclidean symbol using neural networks to render the symbol smooth. This makes the model smooth, mitigating overfitting (Figure 1).

The following section introduces the PDO theory and the proposed model based on the PDO. We also derive the smoothness of the proposed model from the smoothness of neural network.

3 Proposed integral operator : Pseudo-differential integral operator

3.1 Symbol network and PDIO

The primary idea in our study is to parameterize the Euclidean symbol $a(x,\xi)$ using neural networks $a^{nn}_\theta(x,\xi)$. This network is called a *symbol network*. The symbol network $a^{nn}_\theta(x,\xi)$ is assumed to be factorized into $a^{nn}_\theta(x,\xi) = a^{nn}_{\theta_1}(x)a^{nn}_{\theta_2}(\xi)$ (See Appendix C.1). Both smooth functions $a^{nn}_{\theta_1}(x)$ and $a^{nn}_{\theta_2}(\xi)$ are parameterized by fully connected neural networks. We propose a PDIO to approximate the Euclidean PDO using the symbol network and the Fourier transform as follows:

$$\mathcal{K}_a[f](x) := \mathcal{F}^{-1}\left[a^{nn}_\theta(x,\xi)\mathcal{F}[f](\xi)\right] = a^{nn}_{\theta_1}(x)\mathcal{F}^{-1}\left[a^{nn}_{\theta_2}(\xi)\mathcal{F}[f](\xi)\right], \tag{9}$$

where $\mathcal{F}$ is the Fourier transform and $\mathcal{F}^{-1}$ is its inverse. The diagram of the PDIO is explained in Figure 2.

Practically, $\mathcal{F}$ and $\mathcal{F}^{-1}$ in equation 9 are approximated by the FFT, which is an effective algorithm that computes the discrete Fourier transform (DFT). Although the symbol network $a^{nn}_{\theta_2}(\xi)$ is defined on $\mathbb{R}^n$, the inverse DFT is evaluated only on the discrete space $\mathbb{Z}^n$. Therefore, the symbol network $a^{nn}_\theta(x,\xi)$ should be considered on the restricted domain $\mathbb{T}^n \times \mathbb{Z}^n$ (See Appendix C.2 for details). The following section details the definitions and properties of the symbol and PDO on $\mathbb{T}^n \times \mathbb{Z}^n$ to understand the PDIO on the domain $\mathbb{T}^n \times \mathbb{Z}^n$. Moreover, we introduce a theorem that connects between the Euclidean symbol and the restricted Euclidean symbol.

3.2 PDOs on $\mathbb{T}^n \times \mathbb{Z}^n$

Definition 3.1. A *toroidal symbol class* is a set $S^m_{\rho,\delta}(\mathbb{T}^n \times \mathbb{Z}^n)$ consisting of the *toroidal symbols* $a(x,\xi)$, which are smooth in x for all $\xi \in \mathbb{Z}^n$, and satisfy the following inequality:

$$|\triangle^\alpha_\xi \, \partial^\beta_x a(x,\xi)| \le c_{\alpha\beta}\langle\xi\rangle^{m-\rho|\alpha|+\delta|\beta|}, \tag{10}$$

for all $\alpha, \beta \in \mathbb{N}^n_0$, and for all $(x,\xi) \in \mathbb{T}^n \times \mathbb{Z}^n$. Here, $\triangle^\alpha_\xi$ are the difference operators.

The PDO corresponding to the symbol class $S^m_{\rho,\delta}(\mathbb{T}^n \times \mathbb{Z}^n)$ can be defined as follows:

Definition 3.2. The toroidal PDO $T_a : \mathcal{A} \to \mathcal{U}$ with the toroidal symbol $a(x,\xi) \in S^m_{\rho,\delta}(\mathbb{T}^n \times \mathbb{Z}^n)$ is defined by the following equation:

$$T_a(f)(x) = \sum_{\xi \in \mathbb{Z}^n} a(x,\xi)\hat{f}(\xi)e^{2\pi i \xi x}. \tag{11}$$

It is well-known that the toroidal PDO $T_a(f)$ with $f \in C^\infty(\mathbb{T}^n)$ is well defined and $T_a(f) \in C^\infty(\mathbb{T}^n)$ (Ruzhansky & Turunen, 2009).

Here, it is necessary to prove that the restricted symbol network $a^{nn}_\theta|_{\mathbb{T}^n \times \mathbb{Z}^n}$ belongs to a certain toroidal symbol class. To connect the symbol network a^{nn}_θ and the restricted symbol network $a^{nn}_\theta|_{\mathbb{T}^n \times \mathbb{Z}^n}$, we introduce a useful theorem that connects the symbols between the Euclidean symbol and the toroidal symbol.

Theorem 3.3. *(Ruzhansky & Turunen, 2009)* ***(Connection between two symbols)*** *Let $0 < \rho \leq 1$ and $0 \leq \delta \leq 1$. A symbol $\tilde{a} \in S^m_{\rho,\delta}(\mathbb{T}^n \times \mathbb{Z}^n)$ is a toroidal symbol if and only if there exists a Euclidean symbol $a \in S^m_{\rho,\delta}(\mathbb{T}^n \times \mathbb{R}^n)$ such that $\tilde{a} = a|_{\mathbb{T}^n \times \mathbb{Z}^n}$.*

Therefore, it is sufficient to consider whether the symbol network $a^{nn}_\theta(x,\xi)$ belongs to a certain Euclidean symbol class.

3.3 Propositions on the symbol network and PDIO

We show that the symbol network $a^{nn}_\theta(x,\xi)$ with the Gaussian error linear unit (GELU) activation function is contained in a certain Euclidean symbol class using the following proposition:

Proposition 3.4. *Suppose the symbol networks $a^{nn}_{\theta_1}(x)$ and $a^{nn}_{\theta_2}(\xi)$ are fully connected neural networks with nonlinear activation GELU. Then, the symbol network $a^{nn}_\theta(x,\xi) = a^{nn}_{\theta_1}(x)a^{nn}_{\theta_2}(\xi)$ is in $S^1_{1,0}(\mathbb{T}^n \times \mathbb{R}^n)$. Therefore, the restricted symbol network $\widetilde{a^{nn}_\theta} \overset{def}{=} a^{nn}_\theta|_{\mathbb{T}^n \times \mathbb{Z}^n}$ is in a toroidal symbol class $S^1_{1,0}(\mathbb{T}^n \times \mathbb{Z}^n)$.*

Remark 3.5. Here, we focus on the most important case where $\rho = 1$ and $\delta = 0$, since $S^m_{\rho,\delta} \supset S^m_{1,0}$ for $0 < \rho \leq 1$ and $0 \leq \delta < 1$ (Hörmander, 2007). Although the proposition only considers the symbol network with GELU, it can be proved for various activation functions (See Appendix E).

Proof. The fully connected neural network for the symbol network $a^{nn}_{\theta_1}(x)$ is denoted as follows:

$$Z_1^{[l]} = W_1^{[l]}A_1^{[l-1]} + b_1^{[l]} \quad (l = 1, 2, ..., L_1), \quad A_1^{[l]} = \sigma(Z_1^{[l]}) \quad (l = 1, 2, ..., L_1 - 1),$$

where $W_1^{[l]}$ is a weight matrix, $b_1^{[l]}$ is a bias vector in the l-th layer of the network, σ is an element-wise activation function, and $A_1^{[0]} = x$ is an input feature vector, and $Z_1^{[L_1]} = a^{nn}_{\theta_1}(x)$ is an output of the network with $\theta_1 = \{W_1^{[l]}, b_1^{[l]}\}_{l=1}^{L_1}$. Similarly, we define $W_2^{[l]}$, $b_2^{[l]}$, $Z_2^{[l]}$ and $A_2^{[l]}$ $(l = 1, 2, ..., L_2)$ for the neural network $a^{nn}_{\theta_2}(\xi)$.

The neural network $a^{nn}_{\theta_1}(x)$ and its derivative are continuous on a compact set $\mathbb{T}^n$. Therefore, $|\partial^\beta_x a^{nn}_{\theta_1}(x)| \leq c_\beta$ for some constant $c_\beta > 0$ and for all $\beta \in \mathbb{N}^n_0$. For the case $|\alpha| = 0$,

$$|\partial^\alpha_\xi a^{nn}_{\theta_2}(\xi)| = |a^{nn}_{\theta_2}(\xi)| = |W_2^{[L_2]}\sigma(Z_2^{[L_2-1]}) + b_2^{[L_2]}| \leq c_\alpha \langle \xi \rangle, \tag{12}$$

for some constant $c_\alpha > 0$ because the absolute value of GELU $\sigma(z)$ is bounded by linear function $|z|$. Notably,

$$\partial^{e_i}_\xi a^{nn}_{\theta_2}(\xi) = W_1^{[L_2]} diag\left(\sigma'\left(Z_1^{[L_2-1]}\right)\right) \times \cdots \times W_1^{[2]} diag\left(\sigma'\left(Z_1^{[1]}\right)\right) W_1^{[1]} e_i. \tag{13}$$

This result implies that the multi-derivatives of symbol $\partial^\alpha_\xi a^{nn}_{\theta_2}(\xi)$ with $|\alpha| \geq 1$ consists of the product of the weight matrix and the first or higher derivatives of the activation functions. Furthermore, the derivative of GELU is bounded, and the second or higher derivatives of the function asymptotically become zero rapidly, that is, $\sigma^{(k)} \in \mathcal{S}(\mathbb{R})$ when $k \geq 2$ (see Definition E.1). Thus, we have the following inequality:

$$|\partial^\alpha_\xi a^{nn}_{\theta_2}(\xi)| \leq c_\alpha \langle \xi \rangle^{1-|\alpha|}, \tag{14}$$

for all $\alpha \in \mathbb{N}_0^n$ with $|\alpha| \geq 1$ for some positive constants c_α. Therefore, we bound the derivative of the symbol network $a_\theta^{nn}(x, \xi)$ as follows:

$$|\partial_x^\beta \partial_\xi^\alpha a(x, \xi)| = |\partial_x^\beta a_{\theta_1}^{nn}(x)||\partial_\xi^\alpha a_{\theta_2}^{nn}(\xi)| \leq \underbrace{c_\alpha c_\beta}_{=c_{\alpha\beta}} \langle \xi \rangle^{1-|\alpha|}. \tag{15}$$

Therefore, the symbol network $a_\theta^{nn}(x, \xi)$ is in $S_{1,0}^1(\mathbb{T}^n \times \mathbb{R}^n)$ as defined in Definition 2.1. Finally, using Theorem 3.3, we deduce that $\widetilde{a^{nn}} = a^{nn}|_{\mathbb{T}^n \times \mathbb{Z}^n}$ is in $S_{1,0}^1(\mathbb{T}^n \times \mathbb{Z}^n)$. □

We introduce the theorem on the boundedness of a toroidal PDO as follows:

Theorem 3.6. *(Ruzhansky & Turunen, 2009)* ***(Boundedness of a toroidal PDO in the Sobolev space)*** *Let $m \in \mathbb{R}$ and $k \in \mathbb{N}$, which is the smallest integer greater than $\frac{n}{2}$, and let $a : \mathbb{T}^n \times \mathbb{Z}^n \to \mathbb{C}$ such that*

$$|\partial_x^\beta \triangle_\xi^\alpha a(x, \xi)| \leq C \langle \xi \rangle^{m-|\alpha|} \quad \text{for all } (x, \xi) \in \mathbb{T}^n \times \mathbb{Z}^n, \tag{16}$$

and all multi-indices α such that $|\alpha| \leq k$ and all multi-indices β. Then the corresponding toroidal PDO T_a defined in Definition 3.2 extends to a bounded linear operator from the Sobolev space $W^{p,s}(\mathbb{T}^n)$ to the Sobolev space $W^{p,s-m}(\mathbb{T}^n)$ for all $1 < p < \infty$ and any $s \in \mathbb{R}$.

The restricted symbol network $\widetilde{a_\theta^{nn}}$ is in a toroidal symbol class $S_{1,0}^1(\mathbb{T}^n \times \mathbb{Z}^n)$ from Proposition 3.4. Thus, it satisfies the condition in Theorem 3.6. Therefore, the PDIO $\mathcal{K}_a$ (equation 9) with the restricted symbol network $\widetilde{a_\theta^{nn}}$ is a bounded linear operator from $W^{p,s}(\mathbb{T}^n)$ to $W^{p,s-1}(\mathbb{T}^n)$ for all $1 < p < \infty$ and $s \in \mathbb{R}$. This implies that the PDIO is a continuous operator between the Sobolev spaces. Therefore, we expect that the PDIO can be applied to a neural operator (equation 1) to obtain a smooth and general solution operator. A description of its application to neural operator is explained in Section 4.1.

4 NEURAL OPERATOR WITH PDIOS

4.1 PSEUDO-DIFFERENTIAL NEURAL OPERATOR

Using the proposed integral operator (equation 9) with the neural operator (equation 1), the combined model is called a *pseudo-differential neural operator* (PDNO). Consider the general case which the intput function $f_t(x)$ and the output function $f_{t+1}(x)$ are multi-valued function. Let $f_t(x) = [f_{t,i}(x) : \mathbb{R}^n \to \mathbb{R}]_{i=1}^{c_{in}} \in \mathbb{R}^{c_{in}}$ with the number of input channels c_{in} and $x \in \mathbb{R}^n$. Then, the PDIO $\mathcal{K}_a$ is expressed as follows:

$$\mathcal{K}_a(f_t)(x) = \left[\sum_{i=1}^{c_{in}} a_{\theta_1,ij}^{nn}(x) \mathcal{F}^{-1} \left[a_{\theta_2,ij}^{nn}(\xi) \mathcal{F}[f_{t,i}](\xi) \right] (x) \right]_{j=1}^{c_{out}}, \tag{17}$$

where θ_1, θ_2 are the parameters of each symbol network and the c_{out} is the number of output channels with $f_{t+1}(x) \in \mathbb{R}^{c_{out}}$. Indeed, the symbol network has $c_{in} \times c_{out}$ outputs for each channel as displayed in Figure 6. In the experiments, we use three separate symbol networks $a_{\theta_1}^{nn}(x)$, Re $\left(a_{\theta_2}^{nn}(\xi)\right)$, and Im $\left(a_{\theta_2}^{nn}(\xi)\right)$. Each symbol network has input dimension $x, \xi \in \mathbb{R}^n$ and output dimension $c_{in} \times c_{out}$.

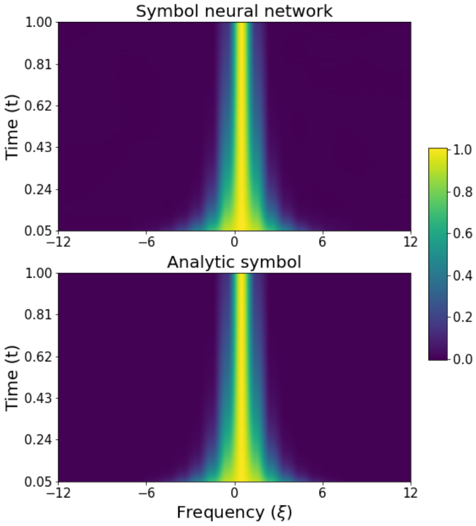

Figure 3: Visualization of the learned symbol from the time dependent PDIO $a_{\theta_1}^{nn}(x,t) a_{\theta_2}^{nn}(\xi,t)$ (top) and analytic symbol $a(x, \xi, t) = e^{-4 \times 0.05 \pi^2 \xi^2 t}$ (bottom) of the solution operator of the 1D heat equation. Note that learned $a_{\theta_1}^{nn}(x,t)$ is a constant function according to x. i.e. $a_{\theta_1}^{nn}(x,t) = c(t)$ (See Figure 7). Therefore, it does not require an x-coordinate to plot the learned symbol.

4.2 TIME DEPENDENT PDIO

Consider the time-dependent PDE

$$\frac{\partial u}{\partial t} = \mathcal{L} u, \quad u(x, 0) = u_0(x), \quad (x, t) \in \mathbb{T}^n \times [0, \infty). \tag{18}$$

This is well-posed and has the unique solution provided that the operator $\mathcal{L}$ is semi-bounded (Hesthaven et al., 2007). The solution is given by $u(x,t) = e^{t\mathcal{L}}u_0(x)$. In the case of the 1D heat equation, $\mathcal{L}$ is $c\partial_{xx}$ with diffusivity constant c. Then, the solution of the heat equation is given by

$$u(x,t) = \sum_{\xi \in \mathbb{Z}} e^{-4\pi^2\xi^2 ct}\hat{u}_0(\xi)e^{2\pi i x\xi}. \tag{19}$$

This shows that the mapping from $u_0(x)$ to $u(x,t)$ is the PDO with the symbol $e^{-4\pi^2\xi^2 ct}$. Consequently, we propose the time dependent PDIOs given as follows:

$$\mathcal{K}_a[f](x,t) := a^{nn}_{\theta_1}(x,t)\mathcal{F}^{-1}\left[a^{nn}_{\theta_2}(\xi,t)\mathcal{F}[f](\xi)\right], \tag{20}$$

where $a^{nn}_{\theta_1}(x,t)$ and $a^{nn}_{\theta_2}(\xi,t)$ are time dependent symbol networks. In the experiment for heat equation, we verify that the time dependent PDIOs approximate time dependent symbol accurately even in a finer time grid than a time grid used for training. Furthermore, the time dependent PDIO is applied to obtain the continuous-in-time solution operator of the PDEs (See experiment on Navier-Stokes equation).

5 Experiments

5.1 Toy example : 1D heat equation

In this experiment, we verify whether the proposed time-dependent PDIO actually learns the symbol of the analytic PDO. We consider the 1D heat equation given in equation 18. The solution operator of the 1D heat equation is a PDO, which is given in equation 19. We aim to learn the mapping from the initial state and time $(u_0(x), t)$ to the solution $u(x,t)$. The initial state $u_0(x)$ is generated from the Gaussian random field $\mathcal{N}(0, 7^4(-\Delta + 7^2)^{-2.5})$ with the periodic boundary conditions. Δ denotes the Laplacian. The diffusivity constant c and the spatial resolution are set to 0.05 and $2^{10} = 1024$, respectively. We use 1000 pairs of train data comprising 10 time grids $t = 0.05 + 0.1n$ ($n = 0, 1, ..., 9$) for each of the 100 initial states. We test for 20 initial states at finer time grids $t = 0.05 + 0.05n$ ($n = 0, 1, ..., 19$). The time dependent PDIO (equation 20) is used and achieves a relative L^2 error lower than 0.01 on both the traning and test sets. Figure 3 shows the symbol network and the analytic symbol given in equation 19 on $(\xi, t) \in [-12, 12] \times [0.05, 1]$. Although the PDIO learns from a sparse time grid, it obtains an accurate symbol for all $t \in [0.05, 1]$.

5.2 Nonlinear solution operators of PDEs

In this section, we verify the PDNO on a nonlinear PDEs dataset. For all the experiments, we use the PDNO that consists of four iterations of the network described in Figure 6 and equation 17 with a nonlinear activation GELU. Fully connected neural networks are used for symbol networks up to layer three and hidden dimension 64. The relative L^2 error is used for the loss function. Detailed hyperparameters are contained in Appendix B. We do not truncate the Fourier coefficient in any of the layers, indicating that we use all of the available frequencies from $-[\frac{s}{2}]$ to $\lceil\frac{s}{2}\rceil - 1$. This is because PDNO does not require additional learning parameters, even if all frequencies are used. However, because evaluations are required at numerous grid points, considerable memory is required in the learning process. In practical settings, it is recommended to truncate the frequency space into appropriate maximum modes k_{max}. We observed that there was little degradation in performance even if truncation was used (See Appendix F.2). All experiments were conducted using up to five NVIDIA A5000 GPUs with 24 GB memory.

Benchmark models We compare the proposed model with the multiwavelet-based model (**MWT**) and the Fourier neural operator (**FNO**), which are the state-of-the-art approaches based on the neural operator architecture. For the difference between **PDNO ($\boldsymbol{a(x,\xi)}$)** and **PDNO ($\boldsymbol{a(\xi)}$)**, see Section 5.3. We conducted the experiments on PDEs, namely Darcy flow and the Navier-Stokes equation. In the case of Navier-Stokes equation, we use the same data attached in Li et al. (2020a). In the case of Darcy flow, we regenerate the data according to the same data generation scheme.

Table 1: Benchmark (relative L^2 error) on Darcy flow on different resolution s.

Resolution	Data	PDNO ($a(x,\xi)$)	PDNO ($a(\xi)$)	MWT Leg	FNO
$s = 32$	train	3.52×10^{-3}	4.08×10^{-3}	1.17×10^{-3}	2.65×10^{-3}
	test	$\mathbf{3.34 \times 10^{-3}}$	3.82×10^{-3}	1.62×10^{-2}	1.78×10^{-2}
$s = 64$	train	2.59×10^{-3}	2.98×10^{-3}	1.81×10^{-3}	2.93×10^{-3}
	test	$\mathbf{2.52 \times 10^{-3}}$	2.85×10^{-3}	1.08×10^{-2}	1.12×10^{-2}
$s = 128$	train	1.58×10^{-3}	2.57×10^{-3}	1.49×10^{-3}	2.77×10^{-3}
	test	$\mathbf{1.62 \times 10^{-3}}$	2.45×10^{-3}	9.27×10^{-3}	1.04×10^{-2}
$s = 256$	train	1.54×10^{-3}	2.62×10^{-3}	1.34×10^{-3}	2.78×10^{-3}
	test	$\mathbf{1.41 \times 10^{-3}}$	2.54×10^{-3}	8.83×10^{-3}	1.01×10^{-2}
$s = 512$	train	1.98×10^{-3}	2.25×10^{-3}	1.32×10^{-3}	2.80×10^{-3}
	test	$\mathbf{1.93 \times 10^{-3}}$	2.17×10^{-3}	9.27×10^{-3}	1.02×10^{-2}

Darcy flow The Darcy flow problem is a diffusion equation with an external force, which describes the flow of a fluid through a porous medium. The steady state of the Darcy flow on the unit box is expressed as follows:

$$\begin{cases} \nabla \cdot (a(x)\nabla u(x)) = f(x), & x \in [0,1]^2 \\ u(x) = 0, & x \in \partial(0,1)^2, \end{cases} \tag{21}$$

where u is density of the fluid, $a(x)$ is the diffusion coefficient, and $f(x)$ is the external force. We aim to learn the nonlinear mapping from $a(x)$ to the steady state $u(x)$, fixing the external force $f(x) = 1$. The diffusion coefficient $a(x)$ is generated from $\psi_{\#}\mathcal{N}(0, (-\Delta + 9I)^{-2})$, where Δ is the Laplacian with zero Neumann boundary conditions, and $\psi_{\#}$ is the pointwise push forward, defined by $\psi(x) = 12$ if $x > 0$, 3 elsewhere. The coefficient imposes the ellipticity on the differential operator $\nabla \cdot (a(x)\nabla)(\cdot)$. We generate $a(x)$ and $u(x)$ using the second order FDM on a 512×512 grid. The lower resolution dataset is obtained by subsampling. We use 1000 train pairs and 100 test pairs and fixed the hyperparameters for all resolutions.

Table 2: Benchmark (relative L^2 error) on the Navier-Stokes equation on the various viscosity ν, the time horizon T, and the number of data N.

Networks	$\nu = 1e-3$ $T = 50$ $N = 1000$	$\nu = 1e-4$ $T = 30$ $N = 1000$	$\nu = 1e-4$ $T = 30$ $N = 10000$	$\nu = 1e-5$ $T = 20$ $N = 1000$
PDNO ($a(x,\xi)$)	0.00903	**0.1320**	0.0679	**0.1093**
PDNO ($a(x,\xi,t)$)	0.0299	0.2296	0.1605	0.1852
MWT Leg	**0.00625**	0.1518	**0.0667**	0.1541
MWT Chb	0.00720	0.1574	0.0720	0.1667
FNO-2D	0.0128	0.1559	0.0973	0.1556
FNO-3D	0.0086	0.1918	0.0820	0.1893

The results on the Darcy flow are presented in Table 1 for various resolutions s. The proposed model achieves the lowest relative error for all resolutions. In the case of $s = 32$, particularly, MWT and FNO exhibit the highest errors. Furthermore, the proposed model maintains its performance even at low resolutions.

Navier-Stokes equation Navier-Stokes equation describes the dynamics of a viscous fluid. In the vorticity formulation, the incompressible Navier-Stokes equation on the unit torus can be expressed as follows:

$$\begin{cases} \frac{\partial w}{\partial t} + u \cdot \nabla w - \nu \Delta w = f, & (x,t) \in (0,1)^2 \times (0,T], \\ \nabla \cdot u = 0, & (x,t) \in (0,1)^2 \times [0,T], \\ w(x,0) = w_0(x), & x \in (0,1)^2, \end{cases} \tag{22}$$

where w is the vorticity, u is the velocity field, ν is the viscosity, and f is the external force. We use the same Navier-Stokes data used in Li et al. (2020a) to learn the nonlinear mapping from $w(x,0), ..., w(x,9)$ to $w(x,10), ..., w(x,T)$, fixing the force $f(x) = 0.1(\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2)))$. The initial condition $w_0(x)$ is sampled from $\mathcal{N}(0, 7^{1.5}(-\Delta + 7^2 I)^{-2.5})$ with periodic boundary conditions. We experiment with four Navier-Stokes datasets:$(\nu, T, N) = (10^{-3}, 50, 1000), (10^{-4}, 30, 1000), (10^{-4}, 30, 10000)$, and $(10^{-5}, 20, 1000)$, where ν is the viscosity, T is the final time to predict, and N is the number of training samples. Notably, the lower the viscosity, the more difficult the prediction. All datasets comprise 64×64 resolutions.

We employ a recurrent architecture to propagate along the time domain. From $w(x,0), ..., w(x,9)$, the model predicts the vorticity at $t = 10$, $\bar{w}(x,10)$. Then, from $w(x,1), ..., w(x,9), \bar{w}(x,10)$, the model predicts the next vorticity $\bar{w}(x,11)$. We repeat this process until $t = T$.

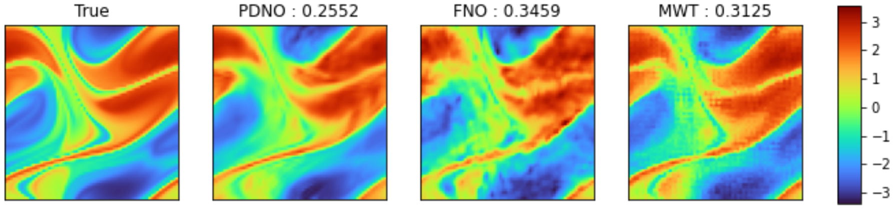

Figure 4: Example of a prediction on the Navier-Stokes data with ν=1e-5 showing the prediction $w(x, 19)$ from inputs $[w(x, 0), ..., w(x, 9)]$. Each value on the top of the figure is the relative L^2 error between the true $w(x, 19)$ and each prediction.

For each experiment, we use 200 test samples. In the case of $(\nu, T, N) = (10^{-3}, 50, 1000)$, we use a batch size 10 or 20 otherwise. Furthermore, we use fixed hyperparameters for the four experiments.

The results on the Navier-Stokes equation are presented in Table 2. In all four datasets, the proposed model exhibits comparable or superior performances. Notably, the relative error improves considerably for $(\nu, T, N) = (10^{-5}, 20, 1000)$, exhibiting the lowest viscosity. Figure 4 displays a sample prediction at $t = 19$, which is highly unpredictable.

5.3 Additional experiments

On Darcy flow, we perform an additional experiment, which does not use a symbol network $a^{nn}_{\theta_1}(x)$, but only use $a^{nn}_{\theta_2}(\xi)$. In this case, the PDNO has the same structure as FNO except for symbol networks. See PDNO ($a(\xi)$) in Table 1. Although less than the original PDNO, the results of the PDNO without the dependency of the x-symbol perform better than the other models, including FNO. This shows why the smoothness of the symbol of PDNO is important. e compare the train and test relative L^2 error on Darcy flow problem in .

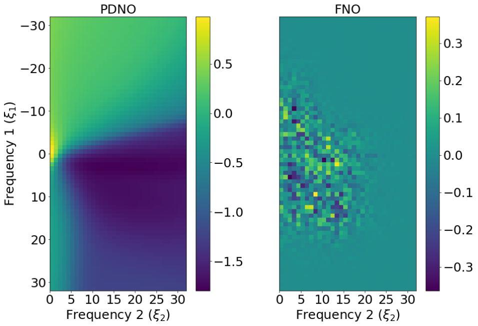

Figure 5: Examples of the real part of learned symbol $a^{nn}_{ij}(\xi)$ from the Navier-Stokes data with $\nu = 1e - 5$. x-axis and y-axis represent frequency domains. As we used real valued functions, the second coordinate is half the first.

On the Navier-Stokes equation, we also show the results with time dependent PDIO in Table 2. This shows a relatively high error, but has the advantage of not using a recursive structure. And, for Navier-Stokes equation with $\nu = 1e - 5$, we compare the train and the test relative L^2 errors along time t in Figure 1. All models show that the test errors grow exponentially according to time t. Among them, PDNO consistently demonstrates the least test errors for all time t. More notable is the difference between the solid lines and the dashed lines, showing that MWT and FNO suffer from overfitting, whereas PDNO does not. The same trend are observed for Darcy flow (See Table 1). This might be related to the smoothness of the symbols of models. Furthermore, the symbols of PDNO and the FNO are visualized in Figure 5.

6 Conclusion

Based on the theory of PDO, we developed a novel PDIO and PDNO framework that efficiently learns mappings between functions spaces. The proposed symbol networks are in a toroidal symbol class that renders the corresponding PDIOs continuous between Sobolev spaces on the torus, which can considerably improve the learning of the solution operators in most experiments. This study revealed an excellent ability for learning operators based on the theory of PDO. However, there is room for improvement in highly complex PDEs such as the Navier Stokes equation, and the time dependent PDIOs are difficult to apply to nonlinear architecture. We expect to solve these problems by using advanced operator theories (Duistermaat, 1996; Hörmander, 1971; Duistermaat & Hörmander, 1972), and the operator learning will solve engineering and physical problems.

REFERENCES

Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Comput. Mech.*, 64 (2):525–545, 2019. ISSN 0178-7675. doi: 10.1007/s00466-019-01740-0. URL https://doi.org/10.1007/s00466-019-01740-0.

Louis Boutet de Monvel. Boundary problems for pseudo-differential operators. *Acta Math.*, 126 (1-2):11–51, 1971. ISSN 0001-5962. doi: 10.1007/BF02392024. URL https://doi.org/10.1007/BF02392024.

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.

R. Courant and D. Hilbert. *Methods of mathematical physics. Vol. I.* Interscience Publishers, Inc., New York, N.Y., 1953.

J. J. Duistermaat. *Fourier integral operators*, volume 130 of *Progress in Mathematics*. Birkhäuser Boston, Inc., Boston, MA, 1996. ISBN 0-8176-3821-0.

J. J. Duistermaat and L. Hörmander. Fourier integral operators. II. *Acta Math.*, 128(3-4):183–269, 1972. ISSN 0001-5962. doi: 10.1007/BF02392165. URL https://doi.org/10.1007/BF02392165.

Weinan E and Bing Yu. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Commun. Math. Stat.*, 6(1):1–12, 2018. ISSN 2194-6701. doi: 10.1007/s40304-018-0127-z. URL https://doi.org/10.1007/s40304-018-0127-z.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323. JMLR Workshop and Conference Proceedings, 2011.

Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 481–490, 2016.

Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. *Advances in Neural Information Processing Systems*, 34, 2021.

Jan S Hesthaven, Sigal Gottlieb, and David Gottlieb. *Spectral methods for time-dependent problems*, volume 21. Cambridge University Press, 2007.

Philipp Holl, Vladlen Koltun, and Nils Thuerey. Learning to control pdes with differentiable physics. *arXiv preprint arXiv:2001.07457*, 2020.

Lars Hörmander. Fourier integral operators. I. *Acta Math.*, 127(1-2):79–183, 1971. ISSN 0001-5962. doi: 10.1007/BF02392052. URL https://doi.org/10.1007/BF02392052.

Lars Hörmander. *The analysis of linear partial differential operators. I.* Classics in Mathematics. Springer-Verlag, Berlin, 2003. ISBN 3-540-00662-1. doi: 10.1007/978-3-642-61497-2. URL https://doi.org/10.1007/978-3-642-61497-2. Distribution theory and Fourier analysis, Reprint of the second (1990) edition [Springer, Berlin; MR1065993 (91m:35001a)].

Lars Hörmander. *The analysis of linear partial differential operators. III.* Classics in Mathematics. Springer, Berlin, 2007. ISBN 978-3-540-49937-4. doi: 10.1007/978-3-540-49938-1. URL https://doi.org/10.1007/978-3-540-49938-1. Pseudo-differential operators, Reprint of the 1994 edition.

Hyung Ju Hwang, Jin Woo Jang, Hyeontae Jo, and Jae Yong Lee. Trend to equilibrium for the kinetic Fokker-Planck equation via the neural network approach. *J. Comput. Phys.*, 419:109665, 25, 2020. ISSN 0021-9991. doi: 10.1016/j.jcp.2020.109665. URL https://doi.org/10.1016/j.jcp.2020.109665.

Rakhoon Hwang, Jae Yong Lee, Jin Young Shin, and Hyung Ju Hwang. Solving pde-constrained control problems using operator learning. *arXiv preprint arXiv:2111.04941*, 2021.

Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric PDE problems with artificial neural networks. *European J. Appl. Math.*, 32(3):421–435, 2021. ISSN 0956-7925. doi: 10.1017/S0956792520000182. URL `https://doi.org/10.1017/S0956792520000182`.

Georgios Kissas, Jacob Seidman, Leonardo Ferreira Guilhoto, Victor M Preciado, George J Pappas, and Paris Perdikaris. Learning operators with coupled attention. *arXiv preprint arXiv:2201.01032*, 2022.

Jae Yong Lee, Jin Woo Jang, and Hyung Ju Hwang. The model reduction of the Vlasov-Poisson-Fokker-Planck system to the Poisson-Nernst-Planck system *via* the deep neural network approach. *ESAIM Math. Model. Numer. Anal.*, 55(5):1803–1846, 2021. ISSN 0764-583X. doi: 10.1051/m2an/2021038. URL `https://doi.org/10.1051/m2an/2021038`.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *arXiv preprint arXiv:2006.09535*, 2020b.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020c.

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

Mohammad Amin Nabian and Hadi Meidani. A deep neural network surrogate for high-dimensional random partial differential equations. *arXiv preprint arXiv:1806.02957*, 2018.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045. URL `https://doi.org/10.1016/j.jcp.2018.10.045`.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Michael Reed and Barry Simon. *Methods of modern mathematical physics. I. Functional analysis*. Academic Press, New York-London, 1972.

Michael Ruzhansky and Ville Turunen. *Pseudo-differential operators and symmetries: background analysis and advanced topics*, volume 2. Springer Science & Business Media, 2009.

Justin Sirignano and Konstantinos Spiliopoulos. DGM: a deep learning algorithm for solving partial differential equations. *J. Comput. Phys.*, 375:1339–1364, 2018. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.08.029. URL `https://doi.org/10.1016/j.jcp.2018.08.029`.

Michael Eugene Taylor. *Pseudodifferential Operators (PMS-34)*. Princeton University Press, 2017.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *arXiv preprint arXiv:2103.10974*, 2021.

Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder-decoder networks for surrogate modeling and uncertainty quantification. *J. Comput. Phys.*, 366:415–447, 2018. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.04.018. URL `https://doi.org/10.1016/j.jcp.2018.04.018`.

Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *J. Comput. Phys.*, 394:56–81, 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2019.05.024. URL `https://doi.org/10.1016/j.jcp.2019.05.024`.

A Notations

We list the main notations throughout this paper in Table 3.

Table 3: Notations

Notations	**Descriptions**
$\mathcal{A}$	an input function space
$\mathcal{U}$	an output function space
$\mathcal{G} : \mathcal{A} \to \mathcal{U}$	an operator from $\mathcal{A}$ to $\mathcal{U}$
$x \in \mathbb{R}^n$ or $\mathbb{T}^n$	a variable in the spatial domain
$\xi \in \mathbb{R}^n$ or $\mathbb{Z}^n$	a variable in the Fourier space
$\hat{f}(\xi)$	Fourier transform of function $f(x)$
$S^m_{\rho,\delta}$	an Euclidean (or toroidal) symbol class
$a(x,\xi) \in S^m_{\rho,\delta}$	an Euclidean (or toroidal) symbol
$a^{nn}_\theta(x,\xi)$	a symbol network parameterized by θ
$T_a : \mathcal{A} \to \mathcal{U}$	a PDO with the symbol $a(x,\xi)$
$\mathcal{K}_a : \mathcal{A} \to \mathcal{U}$	a PDIO with a symbol network $a_\theta(x,\xi)$
$\mathcal{K}_R : \mathcal{A} \to \mathcal{U}$	a Fourier integral operator (Li et al., 2020a)
$\mathcal{F} : \mathcal{A} \to \mathcal{U}, \mathcal{F}^{-1} : \mathcal{U} \to \mathcal{A}$	Fourier transform and its inverse
$\|\xi\|$	Euclidean norm
$\langle\xi\rangle$	$(1+\|\xi\|^2)^{\frac{1}{2}}$
$\triangle^\alpha_\xi$	a difference operator of order α on ξ
k_{max}	the maximum number of Fourier modes

B Hyperparameters

Table 4: Hyperparameters for learning PDNOs on each dataset. # layers, # hidden and activation are for symbol networks.

Data	batch size	learning rate	weight decay	epochs	step size	# channel	# layers	# hidden	activation
Heat equation	20	1×10^{-2}	1×10^{-6}	10000	2000	1	2	40	GELU
Darcy flow	20	1×10^{-2}	1×10^{-6}	1000	200	20	3	32	GELU
Navier-Stokes equation	20	5×10^{-3}	1×10^{-6}	1000	200	20	2	32	GELU

C Analysis on symbol $a(x,\xi)$

C.1 Decomposable assumption $a^{nn}(x,\xi) = a^{nn}(x)a^{nn}(\xi)$

The reason that the symbol is assumed to be decomposable is because of computational costs. With decomposable symbol network $a^{nn}(x)a^{nn}(\xi)$, we need only one IFFT computation

$$\mathcal{K}_a(f)(x) = a^{nn}(x) \underbrace{\sum_{\xi \in \mathbb{Z}^n} a^{nn}(\xi)\hat{f}(\xi)e^{2\pi i \xi x}}_{\text{IFFT}} = a^{nn}(x)\mathcal{F}^{-1}\left[a^{nn}(\xi)\hat{f}(\xi)\right].$$

With the symbol network $a^{nn}(x, \xi)$, an input of IFFT depends on the spatial domain x. Hence, we need IFFT computations as many times as the number of grids in the spatial domain x.

$$\mathcal{K}_a(f)(x) = \underbrace{\sum_{\xi \in \mathbb{Z}^n} a^{nn}(x, \xi)\hat{f}(\xi)e^{2\pi i \xi x}}_{\text{IFFT}} = \mathcal{F}^{-1}\left[a^{nn}(x, \xi)\hat{f}(\xi)\right].$$

Therefore, the non-decomposable symbols require the number of x grid points times the cost of decomposable symbols.

C.2 The reason for considering the symbol network on domain $\mathbb{T}^n \times \mathbb{Z}^n$

In this section, we detail why the proposed model should be addressed in $\mathbb{T}^n \times \mathbb{Z}^n$ instead of $\mathbb{T}^n \times \mathbb{R}^n$. For convenience, we assume that $n = 1$. Let $f : \mathbb{T} \to \mathbb{R}$ and its N points discretization $f(\frac{1}{N}) = y_0$, $f(\frac{2}{N}) = y_1$, ... $f(\frac{N}{N}) = y(1) = y_{N-1}$. Then, the discrete Fourier transform (DFT) of the sequence $\{y_n\}_{0 \leq n \leq N-1}$ is expressed by the following:

$$\xi_k = \frac{1}{N}\sum_{n=0}^{N-1} y_n e^{-2\pi i k \frac{n}{N}} \tag{23}$$

and the inverse discrete Fourier transform (IDFT) of $\{\xi_k\}_{0 \leq k \leq N-1}$ is expressed as follows:

$$y_n = \sum_{k=0}^{N-1} \xi_k e^{2\pi i k \frac{n}{N}}. \tag{24}$$

As N goes ∞, we can see that

$$\lim_{N\to\infty} \xi_k = \lim_{N\to\infty} \frac{1}{N}\sum_{n=0}^{N-1} f(\frac{n+1}{N})e^{-2\pi i k \frac{n}{N}} \to \int_0^1 f(x)e^{-2\pi i k x}\, dx = \hat{f}(k) \tag{25}$$

and

$$\lim_{N\to\infty} y_n = \lim_{N\to\infty} f(\underbrace{\frac{n}{N}}_{x} + \frac{1}{N}) = \lim_{N\to\infty} \sum_{k=0}^{N-1} \xi_k e^{2\pi i k \frac{n}{N}} \to \sum_{k=0}^{\infty} \hat{f}(k)e^{2\pi i k x} = f(x), \tag{26}$$

where $x = \frac{n}{N}$. Thus, DFT is an approximation of integral on $\mathbb{T}$ and IDFT is an approximation of infinity sum on $\mathbb{Z}$. Therefore, the theory of PDO on $\mathbb{T}^n \times \mathbb{Z}^n$ is more suited to our model.

D Resource requirements

Table 5: Resource comparison of PDNO and FNO on NS data. PDNO uses 2 layers symbol network with hidden dimension 32. The memory requirements is obtained from *nvidia-smi* command. We used a single NVIDIA A5000 GPU.

Model	k_{max}	# channels	memory (train)	# parameters	time (sec/epoch)
PDNO	12	20	9939 MB	1.90×10^5	20.77
PDNO	12	30	17819 MB	3.91×10^5	40.73
PDNO	32	20	10143 MB	1.90×10^5	22.05
FNO	12	20	3625 MB	4.66×10^5	4.58
FNO	12	30	3941 MB	1.05×10^6	5.49
FNO	32	20	3957 MB	3.28×10^6	4.80

In Table 5, we compared the memory requirement, the number of parameters and training time of PDNO and FNO. PDNO requires more memory than FNO in training because it needs to compute the symbol networks $a^{nn}(x)$ and $a^{nn}(\xi)$. On the other hand, PDNO has fewer parameters than FNO, so it requires lower storage to save the trained model. If a faster inference is required, the evaluated values of symbol network may be stored. To reduce memory resources during training, one possible future work is to make the symbol network smooth through regularization on the parametric symbol R_{ij} in 3.

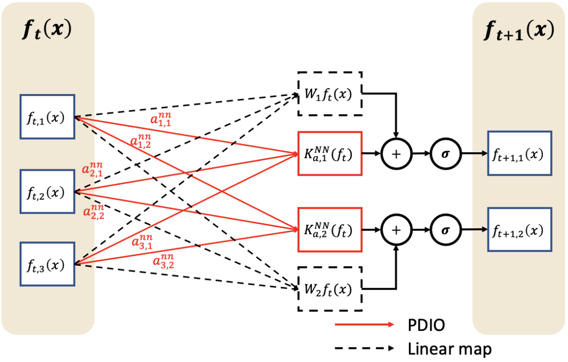

Figure 6: Structure of equation 1 using the integral operator $\mathcal{K}_a$ in equation 17 with $c_{in} = 3$ and $c_{out} = 2$. Each black solid line represents a PDIO with symbol network a_{ij}^{nn}.

E Activation functions for symbol network

In this section, we discuss the activation function for the symbol network. We proved the Proposition 3.4 when GELU activation function is used for the symbol network. Not only GELU, but also other activation functions can be used for the symbol network. To explain this, we first define the Schwartz space (Reed & Simon, 1972) as follows:

Definition E.1. The Schwartz space $\mathcal{S}(\mathbb{R}^n)$ is the topological vector space of functions $f : \mathbb{R}^n \to \mathbb{C}$ such that $f \in C^\infty(\mathbb{R}^n)$ and

$$z^\alpha \partial^\beta f(z) \to 0, \quad \text{as } |z| \to \infty, \tag{27}$$

for every pair of multi-indices $\alpha, \beta \in \mathbb{N}_0^n$.

That is, the Schwartz space consists of smooth functions whose derivatives decay at infinity faster than any power. As mentioned in the proof of Proposition 3.4, it can be easily shown that the second or higher derivatives of GELU is in the Schwartz space $\mathcal{S}(\mathbb{R})$. Because GELU is defined as $\sigma(z) = z\Phi(z)$ with $\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp(-u^2/2)du$, the second or higher derivatives of GELU is the sum of exponential decay functions $\exp(-z^2/2)$. Thus, the second or higher derivatives of the function is in the Schwartz space, that is, $\sigma^{(k)} \in \mathcal{S}(\mathbb{R})$ when $k \geq 2$.

Next, we prove that another activation function $\phi(z)$ is in symbol class $S_{1,0}^1(\mathbb{T}^n \times \mathbb{R}^n)$ if the difference between the function $\phi(z)$ and GELU $\sigma(z)$ is in the Schwartz space. We call a function like $\phi(z)$ a GELU-like activation function. It can be easily shown that the function $\phi(z)$ is bounded by linear function $|z|$ because GELU is bounded by the linear function. Because the Schwartz space is closed under differentiation, $\phi(z) - \sigma(z) \in \mathcal{S}(\mathbb{R})$ implies $\phi^{(k)}(z) - \sigma^{(k)}(z) \in \mathcal{S}(\mathbb{R})$ for $k \in \mathbb{N}$. Because GELU satisfies $\sigma'(z) \leq c_\alpha$ and $\sigma^{(k)} \in \mathcal{S}(\mathbb{R})$ when $k \geq 2$, the activation function $\phi(z)$ also satisfies $\phi'(z) \leq c_\alpha$ and $\phi^{(k)} \in \mathcal{S}(\mathbb{R})$ when $k \geq 2$. Therefore, the proof of Proposition 3.4 can be obtained by changing another activation function $\phi(z)$ instead of GELU $\sigma(z)$. GELU-like activation functions, such as the Softplus (Glorot et al., 2011), and Swish (Ramachandran et al., 2017) etc., satisfy the aforementioned assumption so that it can be used for the symbol network in our PDIO.

We can easily show that the symbol network $a_\theta^{nn}(x, \xi)$ with $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ is in $S_{1,0}^0(\mathbb{T}^n \times \mathbb{R}^n)$. In the proof of Proposition 3.4, we used the characteristic of GELU and its high derivatives. Tanh function is bounded and the first or higher derivatives of tanh function is in the Schwartz space. Therefore, neural network $a_{\theta_2}^{nn}(\xi)$ satisfies the following boundedness:

$$|\partial_\xi^\alpha a_{\theta_2}^{nn}(\xi)| \leq c_\alpha, \quad \text{if } |\alpha| = 0, \tag{28}$$

$$|\partial_\xi^\alpha a_{\theta_2}^{nn}(\xi)| \leq c_\alpha \langle \xi \rangle^{-|\alpha|}, \quad \text{if } |\alpha| \geq 1. \tag{29}$$

Note that the boundedness of the neural network $a_{\theta_1}^{nn}(x)$ is same in the case of GELU. Thus, we can bound the derivative of the symbol network $a_\theta^{nn}(x, \xi)$ as follows:

$$|\partial_x^\beta \partial_\xi^\alpha a(x, \xi)| \leq c_{\alpha\beta} \langle \xi \rangle^{-|\alpha|}. \tag{30}$$

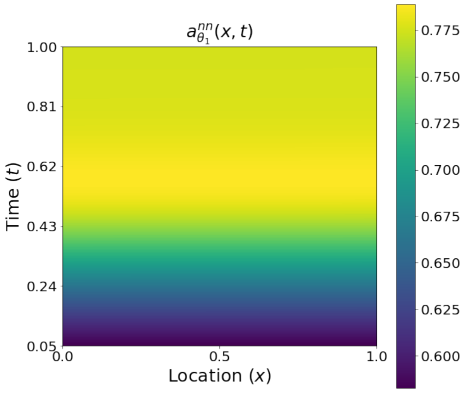

Figure 7: Learned symbol $a^{nn}_{\theta_1}(x,t)$ from 1D heat equation.

Therefore, the symbol network with tanh activation function is in $S^0_{1,0}(\mathbb{T}^n \times \mathbb{R}^n)$. Similarly, it is easy to prove that sigmoid function $\frac{1}{1+e^{-z}}$ also is in a symbol class $S^0_{1,0}(\mathbb{T}^n \times \mathbb{R}^n)$. Therefore, the PDIOs with these two activation functions are bounded linear operators from the Sobolev space $W^{p,s}(\mathbb{T}^n)$ to the Sobolev space $W^{p,s}(\mathbb{T}^n)$ for all $1 < p < \infty$ and any $s \in \mathbb{R}$.

F Additional figures and experiments

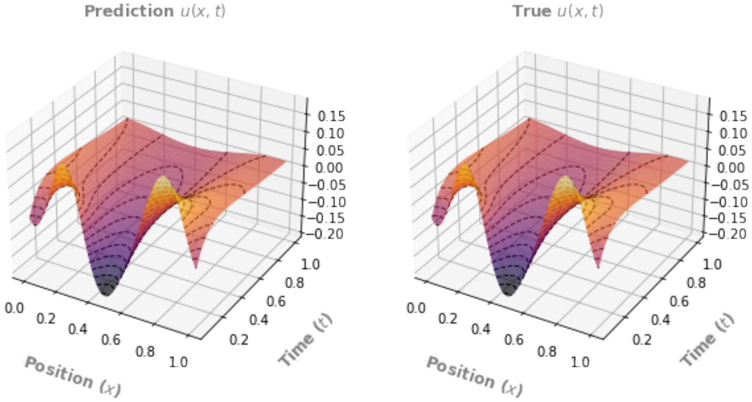

Figure 8: A sample of prediction on 1D heat equation from a PDIO. The model is trained on 1024×10 dataset and evaluated on 1024×20. Dashed lines on the surface are contour lines.

F.1 1D heat equation : Symbol network $a^{nn}_{\theta_1}(x,t)$.

In 1D heat equation experiments, we assume that the symbol is decomposed by $a(x,\xi,t) \approx a^{nn}_{\theta_1}(x,t) \times a^{nn}_{\theta_2}(\xi,t)$. In Figure 7, it shows the learned symbol network $a^{nn}_{\theta_1}(x,t)$ on $(x,t) \in \mathbb{T} \times [0.05,1]$. We can see that $a^{nn}_{\theta_1}(\cdot,t)$ is almost a constant function for each $t \in [0.05,1]$. In this

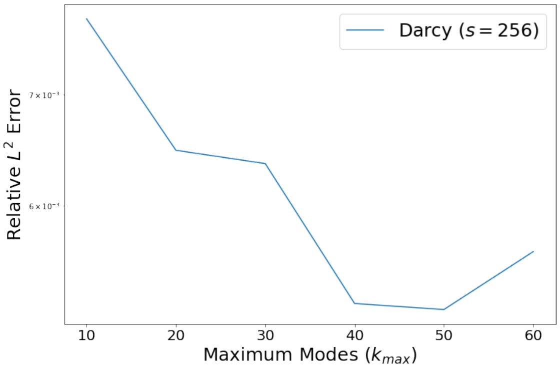

Figure 9: Test relative L^2 error that depends on maximum modes k_{max} of PDNO on Darcy flow (resolution $s = 256$).

respect, $a_{\theta_1}^{nn}(x,t)$ is treated as a function of t by taking the average along x-dimension to visualize $a_{\theta_1}^{nn}(x,t)a_{\theta_2}^{nn}(\xi,t)$ in Figure 3.

In addition, Figure 8 visualizes a sample prediction on 1D heat equation.

F.2 Changes in errors according to k_{max}.

As mentioned in Section 5.2, we use all possible modes. Although PDNO does not require additional parameters to use all modes, it demands more memory in the learning process. So, we perform an additional experiments on Darcy flow by limiting the number of modes k_{max}. In Figure 9, changes in test relative L^2 error along k_{max} are shown. Even with small k_{max}, it still outperforms MWT and FNO (Table 1). And, for $k_{max} \geq 20$, PDNO obtains comparable relative L^2 error on Darcy flow problem.

F.3 Navier-Stokes equation with $\nu = 1e-5$

Samples with the lowest and highest error Figure 11 and Figure 12 show the samples with the highest and the lowest error, respectively. PDNOs consistently obtains the lowest error at all time steps of both samples.

PDNO and FNO with different number of channels We compare the performance of PDNO and FNO, which varies depending on the number of channels. For a fair comparison, the truncation is not used in the Fourier space for both FNO and PDNO. Furthermore, PDNO utilizes only a single symbol network $a_{\theta_2}^{nn}(\xi)$, not $a_{\theta_1}^{nn}(x)$. In Figure 10, as the number of channels increases, the test error decreases in both models. PDNO achieves lower test error than FNO and also shows small gap between the train error and the test error.

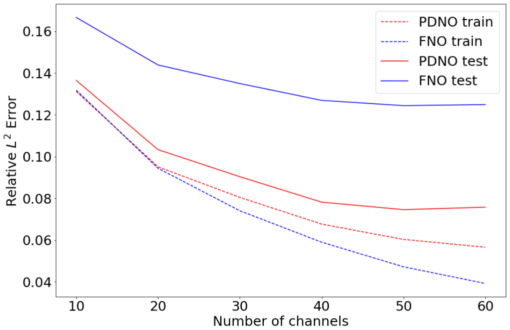

Figure 10: Training and test error of the proposed model and FNO on Navier-Stokes data with $\nu = 1e - 5$ according to the number of channels.

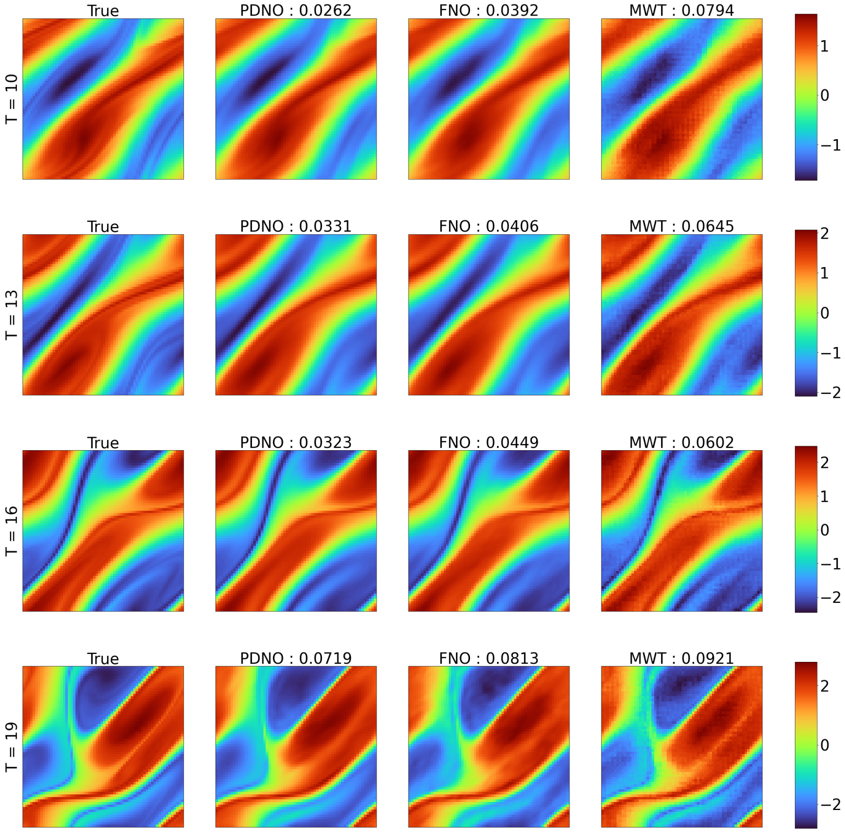

Figure 11: Comparison the prediction on Navier-Stokes equation with $\nu = 1e - 5$. This test sample shows the lowest relative L^2 error on average of three models.

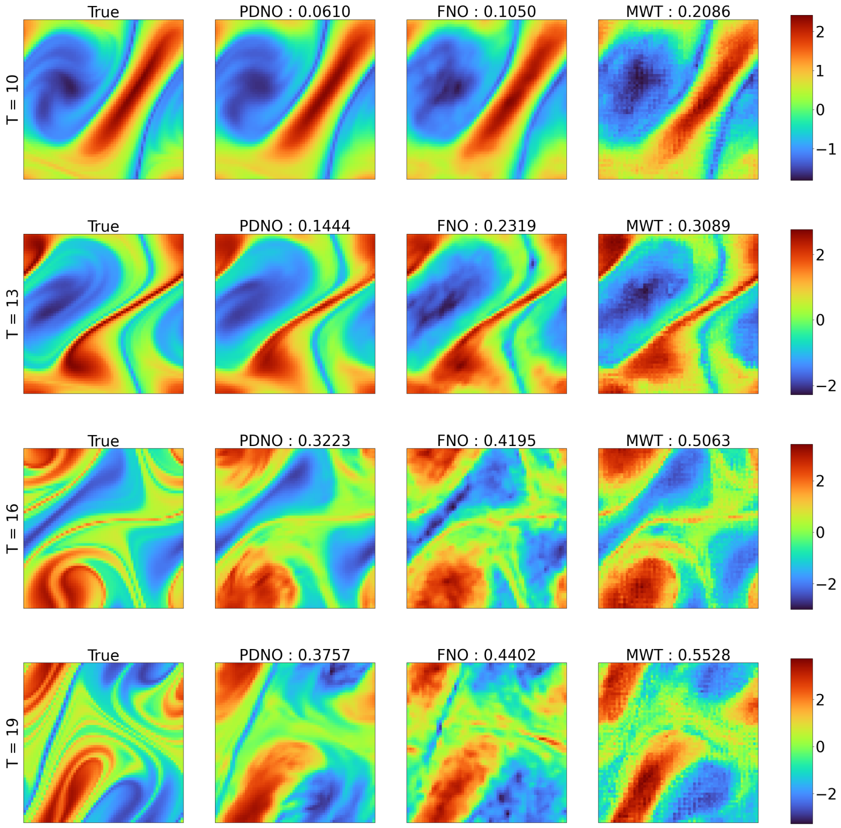

Figure 12: Comparison the prediction on Navier-Stokes equation with $\nu = 1e - 5$. This test sample shows the greatest relative L^2 error on average of three models.