CoNO: Complex Neural Operator for Continuous Dynamical Systems

Karn Tiwari
Department of Electrical Communication Engineering
Indian Institute of Science, Bangalore
Bengaluru, 560012, India
karntiwari@iisc.ac.in

N M Anoop Krishnan
Yardi School of Artificial Intelligence
Indian Institute of Technology, Delhi
New Delhi, 110016, India
krishnan@iitd.ac.in

Prathosh A P
Department of Electrical Communication Engineering
Indian Institute of Science, Bangalore
Bengaluru, 560012, India
prathosh@iisc.ac.in

Abstract

Neural operators extend data-driven models to map between infinite-dimensional functional spaces. These models have successfully solved continuous dynamical systems represented by differential equations, *viz* weather forecasting, fluid flow, or solid mechanics. However, the existing operators still rely on real space, thereby losing rich representations potentially captured in the complex space by functional transforms. In this paper, we introduce a Complex Neural Operator (CoNO), that parameterizes the integral kernel in the complex fractional Fourier domain. Additionally, the model employing a complex-valued neural network along with aliasing-free activation functions preserves the complex values and complex algebraic properties, thereby enabling improved representation, robustness to noise, and generalization. We show that the model effectively captures the underlying partial differential equation with a single complex fractional Fourier transform. We perform an extensive empirical evaluation of CoNO on several datasets and additional tasks such as zero-shot super-resolution, evaluation of out-of-distribution data, data efficiency, and robustness to noise. CoNO exhibits comparable or superior performance to all the state-of-the-art models in these tasks. Altogether, CoNO presents a robust and superior model for modeling continuous dynamical systems, providing a fillip to scientific machine learning.

1 Introduction

Continuous dynamical systems span various scientific and engineering fields, such as physical simulations, molecular biology, climatic modeling, and fluid dynamics, among others [5]. These systems are mathematically represented using PDEs, which are numerically solved to obtain the system's time evolution. The resolution of PDEs necessitates the identification of an optimal solution operator, which maps from functional spaces encompassing initial conditions and coefficients. Achieving this mapping entails discretization procedures for the data capture. Traditionally, numerical methods, such as finite element and spectral methods, have been employed to approximate the solution operator for PDEs. However, these approaches often incur high computational costs and exhibit limited adaptability to arbitrary resolutions and geometries [25, 27].

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Recently, neural operators have shown promise in solving these PDEs in a data-driven fashion [14]. Neural operators extend the neural network to map between infinite dimensional functional space and are a universal approximation of the operator [14]. Operator learning was first proposed by [18], namely, DeepOnet, which theoretically established the universal approximation of operators. DeepONet consists of a branch net and trunk net, where the branch learns the input function operator, and the trunk learns the function space onto which it is projected. Another widely used architecture, Fourier Neural Operator (FNO), a frequency domain method (FDM), was proposed by [17], which consists of uplifting Fourier kernel-based integral using fast Fourier transform and projection blocks. Following this, several frequency transformation-based kernel integral neural operators have been proposed. For instance, [6] introduced spectral methods such as Chebyshev and Fourier series to avoid aliasing error and opaque output in FNO, [30] blends integral kernels with wavelet transformation, which uses time-frequency wavelet localization.

Despite several successes of operator learning for solving PDEs, [2] showed that several problems persist and must be addressed. These include aliasing errors, generalization, robustness to noise, and structure-preserving equivalent operator [2, 6]. Operators must respect the continuous-discrete equivalence while learning the underlying operation, not just a discretized version. This is challenging since the data used for training the operator is provided as a discretized version of the continuous field. As at any finite resolution, the possible mismatch between the continuous-discrete version should not introduce any lead-in error, i.e., this should not lead to any aliasing error [19]. Several operators, such as FNO, suffer from continuous-discrete equivalence, i.e., aliasing error, introduced by pointwise activation.

In the realm of data-driven methods, introducing a learnable order for the fractional discrete Fourier transform [3], which facilitates the seamless integration of features between the time and frequency domains, has been an open research problem. However, this concept becomes less clear when applied to the study of continuous systems, representing a broader, more generalized form of the Fourier transform. Recent research by [23] has significantly addressed the challenge of aliasing errors within the operator framework. Their approach involves the utilization of convolution operators explicitly parameterized in the physical space, diverging from traditional frequency domain methods. Moreover, they have harnessed UNET-based architectures [24] to enhance the architecture's efficiency and memory utilization. It is also worth noting that the study conducted by [26] emphasizes the critical role of overparametrization in achieving superior generalization and optimization performance.

Another aspect that has received less attention is the complex representation of FDMs. The traditional FDMs in operator learning, including FNO, do not perform non-linear transformations on the complex representations of the Fourier transform. Thus, these models do not exploit the rich representations of complex numbers. The allure of complex number representations has grown considerably due to their ability to capture richer information through phase information [21] in complex neural networks [8]. They exhibit advantages such as faster convergence [4, 1] and improved generalization [9]. While [29] highlighted the benefits of combining complex neural networks with their real representation counterparts, their application in the Operator learning framework remains largely unexplored.

Our Contributions. To address these challenges, we present an operator that utilizes Complex neural networks based on frequency domain representations in this work. Specifically, we introduce the Complex Neural Operator (CoNO), a novel deep learning operator designed to establish mappings between infinite-dimensional functional spaces. Table 1 compares several features of CoNO with other existing operators. Our contributions are outlined as follows:

1. **Complex Neural Operator:** CoNO represents the first instance of a Complex Neural Operator that performs operator learning employing a complex neural network.

2. **Fractional Kernel Integral:** CoNO parameterizes the integral kernel within the complex fractional Fourier transform using a single transformation operation with learnable fractional parameters, thereby reducing the number of transformations in comparison to previous operators.

3. **Data efficiency and Robustness to noise:** CoNO demonstrates high generalization even with minimal samples, data instances, and training epochs. Specifically, CoNO gives the same performance as FNO with 1/4 size of the training data. Further, CoNO exhibits improved robustness to noise in the training or testing dataset compared to SOTA operators.

Operators	FDM	Alias Free	Learnable Order	Downscaling	Complex Rep
DeepONet	✗	✗	✗	✗	✗
FNO	✓	✗	✗	✗	✗
WNO	✓	✗	✗	✗	✗
SNO	✓	✓	✗	✗	✗
CoNO (Ours)	✓	✓	✓	✓	✓

Table 1: Comparison of the features of different neural operators with CoNO.

2 Preliminaries

2.1 Operator Learning Framework

Problem Setting: We have followed and adopted the notations in [17]. Let us denote a bounded open set as $D \subset \mathbb{R}^d$, with $A = A(D; \mathbb{R}^{d_a})$ and $U = U(D; \mathbb{R}^{d_u})$ as separable Banach spaces of functions, representing elements in $\mathbb{R}^{d_a}$ and $\mathbb{R}^{d_u}$, respectively. Consider $G^\dagger : A \to U$ to be a nonlinear mapping, arising from the solution operator for a parametric partial differential equation (PDE). It is assumed that there is access to independent and identically distributed observations $(a_j, u_j)_{j=1}^N$, where $a_j \sim \mu$, drawn from the underlying probability measure μ supported on A, and $u_j = G^\dagger(a_j)$.

The objective of operator learning is to construct an approximation for $G^\dagger$ via a parametric mapping $G : A \times \Theta \to U$, or equivalently, $G_\theta : A \to U, \theta \in \Theta$, within a finite-dimensional parameter space Θ. The aim is to select $\theta^\dagger \in \Theta$ such that $G(\cdot, \theta^\dagger) = G_\theta^\dagger \approx G^\dagger$. This framework facilitates learning in infinite dimensional spaces as the solution to the optimization problem in Equation 1 constructed using a with a loss function $L : U \times U \to \mathbb{R}$.

$$\min_{\theta \in \Theta} \mathbb{E}_{a \sim \mu} \left[L(G(a, \theta), G^\dagger(a)) \right], \tag{1}$$

In neural operator learning frameworks, the above optimization problem is solved using a data-driven empirical approximation of the loss function akin to the regular supervised learning approach using train-test observations. Usually, G_θ is parameterized using deep neural networks.

2.2 Complex Neural Networks

In our proposal, we use complex neural networks for approximating G_θ. Here, each neuron will separately output a real and an imaginary part. As demonstrated by [20] [21], complex neural networks often outperform their real-valued counterparts on function approximation tasks. Notably, since they have both real and imaginary parts, they can facilitate learning of mutually orthogonal decision boundaries in the real and imaginary domains, thereby enhancing generalization capabilities. Furthermore, it was observed that critical points in complex neural networks predominantly manifest as saddle points rather than local minima, in contrast to real-valued neural networks [20, 21]. It is noted that stochastic gradient-based optimization algorithms such as SGD can largely avoid saddle points but not local minima [15, 10]. Additionally, complex neural networks exhibit improved training efficiency and enhanced generalization compared to standard CNNs [13].

Note that the activation functions employed in complex neural networks should also respect the complex operations. In our method, we incorporate Complex GeLU (CGeLU) activation functions, which apply independent GeLU [7] [16] functions to a neuron's real and imaginary components, respectively. Formally, this can be expressed as:

$$\text{CGeLU}(z) = \text{GeLU}(\text{Re}(z)) + i\text{GeLU}(\text{Im}(z)). \tag{2}$$

The CGeLU activation function satisfies the Cauchy-Riemann equations when the real and imaginary parts are strictly positive or negative.

2.3 Fractional Fourier transform

In the Operator Learning framework, Operators are defined via architectures comprising functional compositions of integral transforms and nonlinear activation functions in the operator learning framework. For instance, while in DeepONet [18], integral transforms happen in the physical domain, FNO [17] incorporates integral transforms in the frequency domain. In our architecture, we propose to use the Discrete Fractional Fourier Transform (FrFT) with learnable order as the integral transform. This enables learning anywhere 'in between' the physical and the frequency domains. This transform can be of great interest in deep learning due to its ability to capture different types of frequency content and directional features present in data.

Fractional Fourier Transform (FrFT) is a mathematical operation that generalizes the classical Fourier transform by introducing a parameter that controls the transform's degree of rotation [22]. Formally, the FrFT of a function $f(x)$ with respect to the fractional order α is defined as:

$$F_\alpha f(x)(u) = \frac{1}{\sqrt{|2\pi|}} \int_{-\infty}^{\infty} e^{-i\pi\alpha \text{sgn}(t) t^2} f(t) e^{-i2\pi ut}, dt, \tag{3}$$

In the above Equation 3, α is the fractional order, u is the transformed variable, and $\text{sgn}(t)$ is the signum function applied to the variable t.

2.4 Mitigation of Aliasing

The operator learning framework necessitates approximation through non-linear operations, including non-linear pointwise activations, which may introduce arbitrarily high-frequency components into the output signal. The emergence of nonlinearity-induced aliasing can precipitate the symmetry distortion inherent in the physical signal, consequently leading to adverse effects. Moreover, translational invariance, desired in a neural operator, is susceptible to degradation due to aliasing [11, 2].

We employ a two-step process to mitigate aliasing errors within the operator learning paradigm for continuous equivariance. First, before applying any activation function, we upsample the input function at a rate exceeding its frequency bandwidth. Subsequently, we apply a non-linear operation to the upsampled signal, then apply a sinc-based filter [33] followed by downsampling. The sinc-based low-pass filter effectively attenuates higher frequency components in the output signal, thus averting aliasing artifacts and preserving the complex domain information. This approach minimizes aliasing in the operator learning framework, as demonstrated empirically later (Sec. 4.6), maintaining the fidelity and integrity of the physical signal.

3 Complex Neural Operator (CoNO)

In this subsection, we introduce our proposed architecture, Complex Neural Operator CoNO. Our goal is to construct the operator in a structure-preserving manner where we band-limit a function over a given spectrum [32], thus preserving complex continuous-discrete equivalence such that Shannon-Whittaker-Kotel'nikov theorem is obeyed for all continuous operations [31].

Let $a : \mathcal{D}_A \to \mathbb{R}^{d_A}$ denote the input function. We initiate the procedure by applying a point-wise operator P to a and obtaining $v_0 : \mathcal{D}_A \to \mathbb{R}^{d_0}$. The point-wise operator P is parameterized as $P_\theta : \mathbb{R}^{d_A} \to \mathbb{R}^{d_0}$, which operates as $v_0(x) = P_\theta(a(x))$ for $x \in \mathcal{D}_0$, where $\mathcal{D}_0 = \mathcal{D}_A$. Typically, P_θ is realized as a deep neural network. In this paper, we set $d_0 \gg d_A$, designating P as a lifting operator.

Subsequently, we apply the operator Q to $v_0(x)$. This operator is realized as a Complex Convolutional Neural Network (CCNN) with a residual connection, as depicted in Figure 1. This facilitates discretized inversion, thereby maintaining continuous-discrete equivalence through functional interpolation in the complex domain. We compute $v_1 : \mathcal{D}_{v_0} \to \mathbb{R}^{d_1}$ using $v_1(x) = Q_\theta(v_0(x))$ for $x \in \mathcal{D}_{v_0}$, where $Q_\theta : \mathbb{R}^{d_{v_0}} \to \mathbb{R}^{d_{v_1}}$.

Subsequent to this, we employ a complex point-wise operator R on v_1 to obatain $v_2 : \mathcal{D}_1 \to \mathbb{R}^{d_2}$. The point-wise operator R is parameterized as $R_\theta : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$, operating as $v_2(x) = R_\theta(v_1(x))$ for $x \in \mathcal{D}_1$, implemented as a complex deep neural network. In this work, we set $d_2 = d_1$.

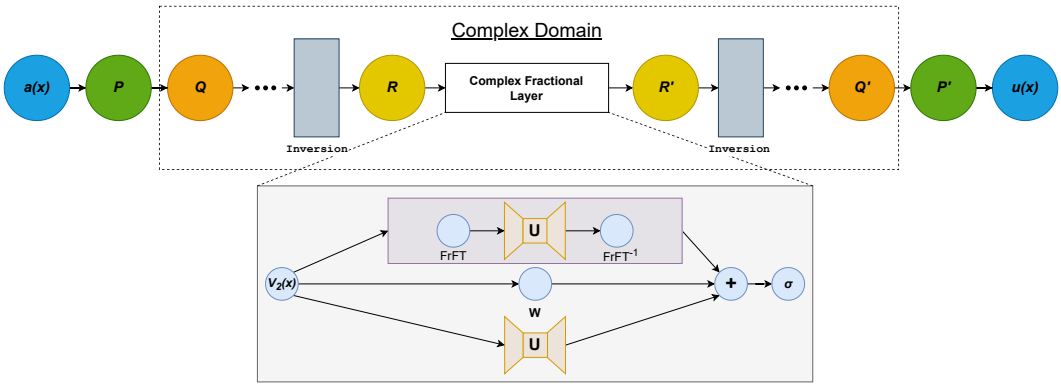

Figure 1: **The full architecture of CoNO.** (1) Input function $a(x)$ is projected into higher dimension through P operation. (2) P is passed through the R operation, which converts the embedding from a real to a complex domain. (3) Lift operation is applied to Q in the complex domain to obtain $v_2(x)$ (4) $v_2(x)$ is passed through a complex fractional integral operator with learnable order parameters where U denotes complex UNET. (5) Then, projection operation Q' is applied to the output. (6) Then it is passed through operation R', which converts the output from the complex to the real domain. (7) Lastly, the operation P' maps to output function u. Inversion denotes the discretization inversion operation on which the complex integral transform is trained during super-resolution.

Following this, we employ a complex fractional nonlinear operator on $v_2(x)$, to obtain $v_3(x)$, which involves a complex UNET-shaped operator with a 3×3 kernel size. During upsampling, zero padding is applied to the signal and convolved with a sinc-based low-pass filter. Downsampling, on the other hand, involves removing the signal components outside the spectrum. This process is implemented using filter-based convolution, accompanied by removing the signal samples at the indices corresponding to padding done during upsampling.

Specifically, $v_3 : \mathcal{D}_{v_2} \to \mathbb{R}^{d_3}$ is calculated as $v_3(x) = Wv_2(x) + K(\alpha;\phi)v_2(x) + K'(\phi)v_2(x)$, where $K(\alpha;\phi)$ and $K'(\phi)$ are kernel integral transformation and convolutional operators, respectively, parameterized by complex neural networks. Here, α represents the fractional complex Fourier transform with a learnable order parameter, and W denotes pointwise complex convolution. In the complex UNET encoding stage, the input function is mapped to vector-valued functions characterized by increasingly contracted domains and higher-dimensional co-domains. Specifically, for each i, we have $\mu(D_i) \geq \mu(D_{i+1})$ and $d_{v_{i+1}} \geq d_{v_i}$. For this to be feasible, in this study, without loss of generality, we adopt the Lebesgue measure μ for μ_i's.

Further, we apply the R' operation to v_3, leading to the computation of $v_4 : \mathcal{D}_3 \to \mathbb{R}^{d_4}$. The point-wise operator R' is parameterized as $R'_\theta : \mathbb{R}^{d_3} \to \mathbb{R}^{d_4}$, acting as $v_4(x) = R_\theta(v_3(x))$ for $x \in \mathcal{D}_3$. Typically, R'_θ is realized as a complex deep neural network. In this paper, we set $d_3 = d_4$.

Lastly, we employ a Complex CNN with a residual connection to transition from the complex domain to the real domain. This results in the computation of $v_5 : \mathcal{D}_{v_4} \to \mathbb{R}^{d_5}$, where $v_5(x) = Q'_\theta(v_4(x))$ for $x \in \mathcal{D}_{v_4}$, and $Q'_\theta : \mathbb{R}^{d_{v_4}} \to \mathbb{R}^{d_{v_5}}$. Finally, we utilize the P' projection operator to map back to the solution domain $u(x)$, resulting in $u : \mathcal{D}_{v_5} \to \mathbb{R}^{d_u}$ with $u(x) = P'_\theta(v_5(x))$ for $x \in \mathcal{D}_{v_5}$. The entire architectural details of CoNO are depicted in Fig. 1.

4 Numerical Experiments and Results

This section presents a comprehensive empirical analysis of CoNO compared to various neural operator baselines, mainly including FDMs, DeepONet[18], FNO[17], Wavelet NO (WNO) [30], and Spectral NO (SNO)[6], on standard datasets. We ensure a diverse selection of partial differential equations (PDEs) taken from [28], encompassing both time-dependent and time-independent problems, to account for the intrinsic computational complexity of the tasks. Further, we evaluate CoNO on several tasks, such as performance on out-of-distribution datasets, data efficiency, and robustness to noise. Finally, we perform ablation studies to understand the contribution of several architectural features in CoNO, such as complex neural network, fractional Fourier transform, aliasing-free

activation function, and bias towards its final performance. All the experiments are conducted on a Linux machine running Ubuntu 20.04.3 LTS on an Intel(R) Core(TM) i9-10900X processor and NVIDIA RTX A6000 GPUs with 48 GB RAM.

4.1 Datasets Description and Baseline Experiments

Setting: In our experiment, we have leveraged a diverse set of partial differential equations (PDEs), encompassing time-independent models, including Burgers and Darcy Flow, and time-dependent models, such as Navier-Stokes, shallow water, and diffusion equations from [28]. This wide-ranging assortment of PDEs has been carefully chosen to facilitate a comprehensive evaluation of the efficacy of our proposed methods. The relative L2 error is presented in Table 2. The descriptions of datasets used in the present work are as follows.

Burger's Dataset: The flow of a viscous fluid in one dimension is modeled by a nonlinear PDE as

$$\frac{\partial u}{\partial t}(x,t) + \frac{\partial}{\partial x}\left(\frac{u^2(x,t)}{2}\right) = \nu \frac{\partial^2 u}{\partial x^2}(x,t), \quad x \in (0,1),, t \in (0,1] \tag{4}$$

This equation, referred to as the 1D Burger's equation, is numerically solved to generate the dataset. This dataset presents the time evolution for one timestep of the equation for a given initial condition. Thus, we aim to learn an operator that maps the initial condition to the next time step. The dataset consists of 2048 training and testing data.

Darcy Flow Dataset: Another widely used dataset, Darcy's equation, represents the flow through porous media. 2D Darcy flow over a unit square is given by

$$\nabla \cdot (a(x)\nabla u(x)) = f(x), \quad x \in (0,1)^2, \tag{5}$$

$$u(x) = 0, \quad x \in \partial(0,1)^2. \tag{6}$$

where $a(x)$ is the viscosity, $f(x)$ is the forcing term, and $u(x)$ is the solution. This dataset employs a constant value of forcing term $F(x) = \beta$. Further, Equation 5 is modified in the form of a temporal evolution as

$$\partial_t u(x,t) - \nabla \cdot (a(x)\nabla u(x,t)) = f(x), \quad x \in (0,1)^2, \tag{7}$$

Thus, the goal on this dataset is to learn the operator that maps the diffusion coefficient to the solution. The dataset consists of 10,000 training and testing data.

Navier Stokes Dataset: 2D Navier-Stokes equation describes the flow of a viscous, incompressible fluid in vorticity form on the unit torus as

$$\partial_t w(x,t) + u(x,t) \cdot \nabla w(x,t) = \nu \Delta w(x,t) + f(x), \quad x \in (0,1)^2,\, t \in (0,T] \tag{8}$$

$$\nabla \cdot u(x,t) = 0, \quad x \in (0,1)^2,\, t \in [0,T] \tag{9}$$

$$w(x,0) = w_0(x), \quad x \in (0,1)^2 \tag{10}$$

where, u represents the velocity field, $w = \nabla \times u$ is the vorticity, w_0 is the initial vorticity, ν is the viscosity coefficient, and f is the forcing function. The goal on this dataset is to learn the operator $G^\dagger$, mapping the vorticity up to time 10 to the vorticity up to some later time $T > 10$. The training and test data in this dataset comprises 5,000 samples with 50 timestamps.

Shallow Water Dataset: Compressible Navier-Stokes equations, that model free-surface flow in the form of hyperbolic PDEs, can be used to model shallow water in 2D as

$$\partial_t h + \nabla \cdot (hu) = 0, \tag{11}$$

$$\partial_t hu + \nabla \cdot \left(u^2 h + \frac{1}{2} g r h^2\right) = -g r h \nabla b, \tag{12}$$

where, u and v denote velocities in the horizontal and vertical directions, h represents water depth, and b characterizes spatially varying bathymetry. The quantity hu corresponds to directional momentum, and g denotes gravitational acceleration. Similar to the previous dataset, the goal on this

dataset is to learn the operator $G^\dagger$, which maps velocity up to time 10 to the vorticity up to some later time $T > 10$. The training and test data comprises 1,000 samples with 101 timestamps.

Diffusion Reaction Equation Dataset: Consider a 2D domain characterized by two non-linearly coupled variables, namely, the activator $u = u(t, x, y)$ and the inhibitor $v = v(t, x, y)$. Assuming that their dynamics are governed by the equations given below

$$\partial_t u = D_u \partial_{xx} u + D_u \partial_{yy} u + R_u, \tag{13}$$
$$\partial_t v = D_v \partial_{xx} v + D_v \partial_{yy} v + R_v, \tag{14}$$

the goal is to learn the operator responsible for mapping the activator's and inhibitor's initial conditions to their respective states later $T > 0$. The training and test data comprises 1,000 samples with 101 timestamps.
Metric: The metric used for experimentation is the mean relative L2 error

$$L = \frac{1}{n} \sum_{j=1}^{n} \frac{\|\hat{u}(j) - u(j)\|_2^2}{\|u(j)\|_2^2} \tag{15}$$

where n is the size of the training data, $u(j)$ represents the j-th ground truth sample of the training data, $\hat{u}(j)$ represents the j-th sample prediction.
Training Details and Baselines: We adhere to standard experimental practices by splitting the dataset into training, testing, and validation sets in ratios of 0.8, 0.1, and 0.1. We employ an ensemble training approach to maintain a level playing field for each operator. This involves specifying a hyperparameter range and randomly selecting a subset of hyperparameters. For the experiments, we use Adam optimizer [12]. We conduct model training for each optimal hyperparameter configuration using random seeds and data splits. And the weight of the best-performing model on the evaluation. Each experiment is repeated three times, and the mean of relative L2 loss is reported.

4.1.1 Comparison with Baselines

First, we evaluate the performance CoNO in comparison to baselines on the five PDE datasets considered. Table 2 shows that CoNO outperforms the baselines on all the datasets except Burgers. We observe among the baselines, FNO outperforms all other models consistently. In Burgers, the performance of FNO and CoNO are comparable with FNO being slightly better. However, it is worth noting that the Burgers dataset corresponds to a 1D equation, which is lower dimensional than all other datasets. These results suggest that CoNO outperforms FNO in more complex and higher dimensional PDEs, while FNO and CoNO may have similar performance in simpler, low dimensional PDEs.

Datasets	DeepONet	FNO	WNO	SNO	CoNO(Ours)
Burgers	$0.027_{\pm 0.002}$	$0.021_{\pm 0.003}$	$0.032_{\pm 0.001}$	$0.23_{\pm 0.01}$	$0.022_{\pm 0.002}$
Darcy	$0.028_{\pm 0.001}$	$0.024_{\pm 0.003}$	$0.054_{\pm 0.004}$	$0.61_{\pm 0.02}$	$0.021_{\pm 0.003}$
Navier Stokes	$0.65_{\pm 0.02}$	$0.41_{\pm 0.02}$	$0.73_{\pm 0.01}$	$8.4_{\pm 0.2}$	$0.36_{\pm 0.01}$
Shallow Water	$0.0064_{\pm 0.0003}$	$0.00049_{\pm 0.00004}$	$0.0074_{\pm 0.0005}$	$0.032_{\pm 0.001}$	$0.00047_{\pm 0.00003}$
Diffusion	$0.92_{\pm 0.01}$	$0.91_{\pm 0.02}$	$0.95_{\pm 0.02}$	$7.3_{\pm 0.1}$	$0.89_{\pm 0.01}$

Table 2: Relative L2 error of CoNO and other baselines for different PDE datasets. The best result is highlighted in blue and the second best in orange.

4.2 Zero Shot Superresolution

The neural operator exhibits mesh invariance, allowing it to undergo training on lower-resolution data and subsequently be applied to higher-resolution data, thereby achieving zero-shot superresolution. CoNO has the capability for zero-shot superresolution, while among the baselines only FNO can perform zero-shot superresolution. To this extent, the neural operators were trained on a 128×128 resolution and tested on higher and lower resolutions for the Darcy flow dataset. Table 3, presents the relative L2 error of CoNO and FNO on superresolution. We note that CoNO outperforms FNO significantly on all the resolutions. This analysis concludes that the L2 error of CoNO does not grow significantly compared to FNO across varying resolutions.

Resolution	DeepONet	FNO	WNO	SNO	CoNO(Ours)
85x85	-	$0.052_{\pm 0.002}$	-	-	$0.044_{\pm 0.001}$
128x128	$0.028_{\pm 0.002}$	$0.024_{\pm 0.002}$	$0.054_{\pm 0.002}$	$0.61_{\pm 0.02}$	$0.021_{\pm 0.002}$
256x256	-	$0.039_{\pm 0.002}$	-	-	$0.029_{\pm 0.002}$
512x512	-	$0.058_{\pm 0.002}$	-	-	$0.043_{\pm 0.002}$

Table 3: Relative L2 error for zero-shot superresolution by CoNO and FNO. Note that all the models are trained on 85×85 resolution and tested on other resolutions. The best result is highlighted in blue and the second best in orange.

4.3 Out-of-Distribution Generalization

In this study, we conducted experiments on the Darcy flow dataset, where during training, we set the force term f to a constant value of $\beta = 1.0$. Subsequently, we evaluated the trained model on various values of β to assess its out-of-distribution generalization capabilities, as illustrated in Table 4. Remarkably, our results consistently demonstrate that CoNO exhibits superior generalization performance compared to other operators.

Beta Coeff	DeepONet	FNO	WNO	SNO	CoNO (Ours)
0.01	$0.80_{\pm 0.01}$	$0.79_{\pm 0.03}$	$0.80_{\pm 0.03}$	$10.80_{\pm 0.2}$	$0.76_{\pm 0.01}$
1.0	$0.028_{\pm 0.001}$	$0.024_{\pm 0.002}$	$0.054_{\pm 0.003}$	$0.61_{\pm 0.02}$	$0.021_{\pm 0.001}$
10.0	$0.068_{\pm 0.001}$	$0.068_{\pm 0.002}$	$0.084_{\pm 0.003}$	$0.54_{\pm 0.02}$	$0.067_{\pm 0.001}$
100.0	$0.075_{\pm 0.001}$	$0.074_{\pm 0.002}$	$0.089_{\pm 0.003}$	$0.53_{\pm 0.02}$	$0.072_{\pm 0.001}$

Table 4: Zero shot out of distribution generalization where we trained on beta coeff 1.0 for Darcy Flow and tested for other beta coeff for different Neural Operators. The best result is highlighted in blue and the second best in orange.

4.4 Data Efficiency

Now, we evaluate the data efficiency of CoNO in comparison to other baselines. Specifically, we conducted experiments with various training splitting ratios, ranging from 0.8 to 0.2, to investigate data efficiency. All the experiments are performed on the Darcy flow dataset with a forcing term $\beta = 1.0$. Our findings reveal that CoNO consistently outperforms alternatives across all splitting ratios. Notably, with just 20% of the training data, CoNO performs equivalent to FNO (see Tab. 5).

Ratio	DeepONet	FNO	WNO	SNO	CoNO(Ours)
0.8	$0.028_{\pm 0.001}$	$0.024_{\pm 0.002}$	$0.054_{\pm 0.003}$	$0.61_{\pm 0.02}$	$0.021_{\pm 0.001}$
0.6	$0.029_{\pm 0.01}$	$0.025_{\pm 0.001}$	$0.054_{\pm 0.003}$	$0.64_{\pm 0.02}$	$0.021_{\pm 0.001}$
0.4	$0.031_{\pm 0.001}$	$0.025_{\pm 0.002}$	$0.057_{\pm 0.003}$	$0.65_{\pm 0.02}$	$0.022_{\pm 0.001}$
0.2	$0.034_{\pm 0.001}$	$0.028_{\pm 0.002}$	$0.061_{\pm 0.003}$	$0.67_{\pm 0.02}$	$0.024_{\pm 0.001}$

Table 5: Data Efficiency for the different ratio of dataset which is used for training for different Neural Operators. The best result is highlighted in blue and the second best in orange.

4.5 Robustness to Noise

In this study, we performed experiments introducing different noise levels into the training and testing datasets using Gaussian noise. The noise addition process can be explained as follows: For each input sample denoted as $x(n)$ within the dataset D, we modified it by adding Gaussian noise with parameters $\gamma N(0, \sigma_D^2)$. Here, σ_D^2 represents the variance of the entire dataset, and γ indicates the specified noise intensity level.

Further, the models were trained on pristine and noisy data with 1% and 5% noise. These models were then cross-evaluated again on pristine data, and noisy data with 1% and 5% noise, covering all combinations. Our investigation yielded notable results, particularly when evaluating the performance

of CoNO in the presence of noise within both the training and testing datasets (see Tab. 6). Remarkably, CoNO exhibits consistent performance irrespective of the presence of noise. Specifically, the noisy training + testing yielded the same result as the pristine dataset confirming the robustness of CoNO to noisy dataset.

Setting	γ %	DeepONet	FNO	WNO	SNO	CoNO (Ours)
	-	$0.028_{\pm 0.001}$	$0.024_{\pm 0.003}$	$0.054_{\pm 0.004}$	$0.61_{\pm 0.02}$	$0.021_{\pm 0.003}$
Noisy Training	1%	$0.029_{\pm 0.003}$	$0.025_{\pm 0.003}$	$0.054_{\pm 0.004}$	$0.61_{\pm 0.02}$	$0.022_{\pm 0.003}$
	5%	$0.030_{\pm 0.003}$	$0.026_{\pm 0.003}$	$0.055_{\pm 0.004}$	$0.62_{\pm 0.02}$	$0.023_{\pm 0.003}$
Noisy Testing	1%	$0.032_{\pm 0.003}$	$0.028_{\pm 0.003}$	$0.054_{\pm 0.003}$	$0.64_{\pm 0.03}$	$0.022_{\pm 0.003}$
	5%	$0.033_{\pm 0.003}$	$0.028_{\pm 0.003}$	$0.054_{\pm 0.003}$	$0.65_{\pm 0.03}$	$0.022_{\pm 0.003}$
Noisy Training + Testing	1%	$0.030_{\pm 0.003}$	$0.025_{\pm 0.003}$	$0.055_{\pm 0.004}$	$0.62_{\pm 0.02}$	$0.021_{\pm 0.003}$
	5%	$0.032_{\pm 0.003}$	$0.025_{\pm 0.003}$	$0.055_{\pm 0.004}$	$0.62_{\pm 0.02}$	$0.021_{\pm 0.003}$

Table 6: Robustness to Gaussian noise during training and testing for different settings for different Neural Operator. The best result is highlighted in blue and the second best in orange.

4.6 Ablation and Comparison

In order to gain insight into the CoNO architecture and how different components impact the performance, we perform ablation studies. All ablation experiments were conducted for the Darcy flow dataset with beta coeff 1.0. **Vanilla CoNO Architecture:** First, we started with the FNO architecture, where Fourier layers are fused into one layer, and a complex neural network is used in a complex domain. Our findings demonstrate that this configuration consistently performs at a level equivalent to FNO as evident from Table 7. **Fractional Fourier Transform Experiment:** In an alternative experiment, we replaced the CoNO transformation layer with a Fourier transform. This modification resulted in a slightly diminished performance compared to the CoNO architecture as evident from Table 7. Also, in CoNO with the Fractional Fourier transform layer, we found that order along different dimensions after training was 0.98 and 0.97, respectively.

Analysis of Alias-Free Activation: We conducted experiments to assess the impact of alias-free activation within our proposed architecture. The results indicate that the absence of alias-free activation leads to degradation in performance of the CoNO as depicted in Table 7. **Effect of Bias Removal:** In a separate investigation, we removed the W and U components from the CoNO model. This adjustment decreased the model's performance, strongly indicating that bias also plays an important role in the learning dynamics of the system.

	Relative L2 error
Vanilla CoNO	$0.024_{\pm 0.002}$
CoNO - FrFT	$0.024_{\pm 0.001}$
CoNO - Alias Free	$0.022_{\pm 0.003}$
CoNO - Bias	$0.026_{\pm 0.001}$
CoNO	$0.021_{\pm 0.001}$

Table 7: Ablation results to study the impact of different components on CoNO.

5 Concluding Insights

Altogether, we present a novel operator learning paradigm, namely Complex Neural Operator (CoNO), that leverages complex neural networks and the complex fractional Fourier transform as an integral operator, thereby ensuring continuous equivalence. This work demonstrates that the rich representation of complex neural networks can be exploited in the operator learning paradigm to develop robust, data-efficient, and superior neural operators that can learn the function-to-function maps in an improved fashion. CoNO outperforms existing operators in terms of performance, zero-shot superresolution, out-of-distribution generalization, and robustness to noise. CoNO, thus, paves the way for creating efficient operators for inferring real-time partial differential equations (PDEs).

Limitations and future work. Although not demonstrated empirically, the architecture of CoNO is capable of effectively downscaling and upscaling the output. Thus, CoNO can also be trained

with differing input and output resolutions. However, the performance of CoNO upon upscaling/downscaling requires further investigation. To further advance our understanding of CoNO, it is crucial to delve into the underlying mathematical and algorithmic principles. Specifically, we need to unravel the learning mechanisms within the latent space and provide the theoretical foundation of complex operators. Furthermore, our research presents novel challenges that warrant investigation. These include tackling the initialization procedures for fractional orders, devising streamlined architectures for complex neural operators, delving into the creation of equivariant complex operators, and elucidating the crucial role played by the fractional Fourier transform in the acquisition of insights into the continuous dynamics of complex systems. These can be pursued as part of future studies.

References

[1] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International conference on machine learning*, pages 1120–1128. PMLR, 2016.

[2] Francesca Bartolucci, Emmanuel de Bézenac, Bogdan Raonić, Roberto Molinaro, Siddhartha Mishra, and Rima Alaifari. Are neural operators really neural operators? frame theory meets operator learning. *arXiv preprint arXiv:2305.19913*, 2023.

[3] Cagatay Candan, M Alper Kutay, and Haldun M Ozaktas. The discrete fractional fourier transform. *IEEE Transactions on signal processing*, 48(5):1329–1337, 2000.

[4] Ivo Danihelka, Greg Wayne, Benigno Uria, Nal Kalchbrenner, and Alex Graves. Associative long short-term memory. In *International conference on machine learning*, pages 1986–1994. PMLR, 2016.

[5] Lokenath Debnath and Lokenath Debnath. *Nonlinear partial differential equations for scientists and engineers*. Springer, 2005.

[6] Vladimir Fanaskov and Ivan Oseledets. Spectral neural operators. *arXiv preprint arXiv:2205.10573*, 2022.

[7] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[8] Akira Hirose. *Complex-valued neural networks*, volume 400. Springer Science & Business Media, 2012.

[9] Akira Hirose and Shotaro Yoshida. Generalization characteristics of complex-valued feedforward neural networks in relation to signal coherence. *IEEE Transactions on Neural Networks and learning systems*, 23(4):541–551, 2012.

[10] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *International conference on machine learning*, pages 1724–1732. PMLR, 2017.

[11] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Manny Ko, Ujjawal K Panchal, Héctor Andrade-Loarca, and Andres Mendez-Vazquez. Coshnet: A hybird complex valued neural network using shearlets. *arXiv preprint arXiv:2208.06882*, 2022.

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

[15] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *Conference on learning theory*, pages 1246–1257. PMLR, 2016.

[16] Minhyeok Lee. Gelu activation function in deep learning: A comprehensive mathematical analysis and performance. *arXiv preprint arXiv:2305.12073*, 2023.

[17] 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*, 2020.

[18] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[19] Hagay Michaeli, Tomer Michaeli, and Daniel Soudry. Alias-free convnets: Fractional shift invariance via polynomial activations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16333–16342, 2023.

[20] Tohru Nitta. On the critical points of the complex-valued neural network. In *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP'02.*, volume 3, pages 1099–1103. IEEE, 2002.

[21] Tohru Nitta. *Complex-valued neural networks: Utilizing high-dimensional parameters: Utilizing high-dimensional parameters*. IGI global, 2009.

[22] Haldun M Ozaktas, Orhan Arikan, M Alper Kutay, and Gozde Bozdagt. Digital computation of the fractional fourier transform. *IEEE Transactions on signal processing*, 44(9):2141–2150, 1996.

[23] Bogdan Raonić, Roberto Molinaro, Tobias Rohner, Siddhartha Mishra, and Emmanuel de Bezenac. Convolutional neural operators. *arXiv preprint arXiv:2302.01178*, 2023.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[25] Granville Sewell. *Analysis of a finite element method: PDE/PROTRAN*. Springer Science & Business Media, 2012.

[26] Muhammad Shafiq and Zhaoquan Gu. Deep residual learning for image recognition: A survey. *Applied Sciences*, 12(18):8972, 2022.

[27] Pavel Ŝolín. *Partial differential equations and the finite element method*. John Wiley & Sons, 2005.

[28] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

[29] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks (2017). *arXiv preprint arXiv:1705.09792*, 2017.

[30] Tapas Tripura and Souvik Chakraborty. Wavelet neural operator: a neural operator for parametric partial differential equations. *arXiv preprint arXiv:2205.02191*, 2022.

[31] Michael Unser. Sampling-50 years after shannon. *Proceedings of the IEEE*, 88(4):569–587, 2000.

[32] M Vetterli, J Kovacevic, and VK Goyal. Foundations of signal processing, cambridge university press, cambridge, 2014.

[33] Leonid P Yaroslavsky. Fast signal sinc-interpolation methods for signal and image resampling. In *Image Processing: Algorithms and Systems*, volume 4667, pages 120–129. SPIE, 2002.