A Solution Method for Partial Differential Equations by Fusing Fourier Operators and U-Net

Tao Zhang * and Xinran Zhu

Posted Date: 25 February 2025

doi: 10.20944/preprints202502.1895.v1

Keywords: neural operator; Partial differential equation; U-Net; Fourier transform

 doi:10.20944/preprints202502.1895.v1

Article

A Solution Method for Partial Differential Equations by Fusing Fourier Operators and U-Net

Tao Zhang * and Xinran Zhu

School of Information and Mathematics, Yangtze University, Jingzhou 434023, China
* Correspondence: tzhang@yangtzeu.edu.cn; Tel.: 18986660997

Abstract: In scientific and engineering computations, the efficient solution of partial differential equations (PDEs) is of great significance. This paper proposes an innovative method based on the combination of the U-Net neural network and the Fourier neural operator, aiming to improve the accuracy and efficiency of solving PDEs. The unique encoder-decoder structure and skip connections of the U-Net neural network can effectively extract and integrate spatial domain features, accurately depicting the spatial structure of PDEs. The Fourier neural operator, by means of the Fourier transform, deeply explores and processes features in the frequency domain, capturing the frequency characteristics of the solutions of PDEs. By organically combining the two, this method not only preserves spatial details but also makes full use of frequency domain information. It significantly reduces the model's dependence on large scale data and enhances its generalization ability. Experimental results show that compared with traditional methods, this method performs outstandingly in various complex partial differential equation(PDE) solving tasks, achieving higher accuracy. It provides a new and highly promising solution for the solution of PDEs and is expected to be widely applied in related fields.

Keywords: neural operator; Partial differential equation; U-Net; Fourier transform

1. Introduction

In fluid mechanics, geophysics, biology, and many other fields, the partial differential equation (PDE) model serves as a crucial tool for characterizing various phenomena. Related problems encompass physical model building, the solution of partial differential equations, and parameter inversion. The process of solving a partial differential equation involves obtaining a spatially accurate solution or a numerical solution under the known partial differential equation and other constraints [1–4].

Traditional numerical solution methods for partial differential equations rely on discrete scheme based finite difference systems, commonly referred to as the computational fluid dynamics method (CFD). Examples include the Runge-Kutta method, the prediction correction method, the finite element method [5], the finite difference method [6], and the finite volume method [7], among others. In contrast to traditional solving methods, neural networks exhibit distinct advantages in solving differential equations. Neural networks possess a powerful data fitting capacity, and even a small scale neural network can handle simple partial differential equations [8]. Employing the neural network approach to solve differential equations effectively addresses issues such as poor adaptability and conditional limitations of grid sections. Compared with traditional numerical methods, the neural network method requires fewer sampling points to achieve higher accuracy. Moreover, once a neural network model is trained, it can be directly utilized to calculate the numerical results at any point within the obtained domain. Since the 1990s, some scholars have initiated research on the mathematical foundations and methods of using neural networks to solve differential equations. In 1990, Hornik et al. [9] demonstrated that under certain conditions, a multilayer neural network can approximate any function and its derivative, even when the function only has a generalized derivative. This finding laid the groundwork for neural network based solutions to partial differential equations. In 1998, Lagaris and Likas [10] put forward a method for solving the first boundary value problem using an

 doi:10.20944/preprints202502.1895.v1

artificial neural network (ANN). In 2011, Kumar and Yadav [11] investigated multilayer neural network (MLP) and radial basis function neural network (RBF) models for solving differential equations, and compared and summarized the applications of MLP and RBF in this regard. In 2018, Winovich et al. [12] proposed a quantitative uncertainty convolutional neural network (ConvPDE-UQ) for the problem of heterogeneous elliptic partial differential equations on different domains. Also in 2018, Long and Dong et al. [13,14] introduced a novel feed forward neural network (PDE - Net). Its core concept is to approximate the differential operator with a convolution kernel, construct a network to approximate the nonlinear partial differential equation system, and achieve long term prediction of its solution. A convolutional network is essentially an input to output mapping, capable of learning numerous mapping relationships between input and output without the need for any precise mathematical expressions between them. However, due to the increase in dimension, the solution accuracy of the convolutional neural network model [15] is slightly diminished. Most of the above mentioned neural network based methods for solving partial differential equations are focused on the initial boundary value problem. The network takes space coordinates or time variables as inputs and outputs the solution values in space. When constructing solutions by integrating partial differential equations with neural networks, the optimization objectives typically consist of two parts: the errors at the initial - solution and boundary value sampling points, and the errors at the sampling points within the domain subject to the differential equation constraints.

In addition, when we need to rapidly obtain accurate solutions for a set of partial differential equations, one approach we can adopt is the neuronal operator method. This method can be learned based on the mapping from the source-term function to the solution function of the partial differential equation set [16].

The Fourier transform holds a significant position in deep learning models. It aids in the proof of the universal approximation theorem [17] and can also accelerate the training of convolutional neural networks [18]. Keivan Alizadeh Vahid et al. [19] proposed extending the butterfly operation in the traditional fast Fourier transform and applying it to the design of convolutional neural network architectures. They demonstrated that the computational complexity of the alternative network layer through butterfly transformation was reduced from N to $N \log N$. Experiments also confirmed that the proposed scheme effectively enhanced the model's accuracy. Integrating the Fourier transform into the neural network model can effectively decrease the model's computational complexity. Moreover, the physical trends in the data can be effectively captured through the Fourier transform, which offers great advantages for solving partial differential equations. Li [20] proposed the neural operator method, which can be evaluated at any point in time and in any space. Additionally, a general approximation theorem for neural operators was summarized and proven. Li [21] put forward the Fourier neural operator, where the integral operator operates directly in the Fourier space through parameterization. Evidently, the neural operator approach is a fully data-driven method, necessitating a large number of high-quality datasets. However, in the analysis of complex physical, biological, or engineering systems, the cost of data acquisition is often prohibitively high. As a result, we inevitably encounter the challenge of drawing long-term conclusions with partial information, making these methods inefficient. Kushnure et al. [22] proposed an architecture named MS U-Net for multi-scale feature representation and recalibration. This architecture uses the Res2Net module to retain the contextual information of segmented objects, and it squeezes and motivates the network to recalibrate multi-scale feature channels, thereby enhancing the network's ability to describe high-level features. In the literature [23], for the same data preprocessing, the feature representations of each modality learned by the network are fused at later stages. In the last layer of convolution, a larger-scale feature map is connected to achieve the communication of contextual information between feature maps of different sizes.

This integration of multi-scale feature representation and recalibration mechanisms allows the network to better capture both local and global information, which is crucial for solving complex partial differential equations. By leveraging these advanced techniques, the model can achieve higher accuracy and efficiency in solving PDEs, even in scenarios where data is limited or incomplete.

 doi:10.20944/preprints202502.1895.v1

In summary, the combination of Fourier transforms, neural operators, and multi-scale feature recalibration provides a powerful framework for solving partial differential equations. These methods not only reduce computational complexity but also enhance the model's ability to capture physical trends and high-level features, making them highly effective for a wide range of applications in science and engineering.

The model extracts information features in the frequency domain through Fourier transform, capturing the key information of the equations, and utilizes convolution neural networks to extract the high-level internal features of the system. By employing a connection-based mechanism for reducing the number of network layers, the model can efficiently solve partial differential equations (PDEs) from both the global and internal characteristics of the system. U-Net extracts image features in the form of convolution, whose essence lies in computing the derivatives of image pixels to capture the variations in the image. On the other hand, the Fourier neural operator calculates derivatives in the form of filter operators, enabling rapid computation through convolution functions.

Building on this foundation, this paper proposes a Fourier neural operator-based U-Net neural network model. This model approximates higher-order differential operators using multiple low-order constrained convolution kernels, thereby better characterizing the properties of the differential equations. In the model design, the feed forward neural operator serves as the core component, containing tunable parameters that can be continuously optimized through training. This design not only enhances the model's solving accuracy but also supports long-term stable numerical solutions, as validated by experimental results.

2. Preliminaries

2.1. Parametric Partial Differential Equations

Let G be the nonlinear operator, general parametric PDEs can be expressed as: $G(u, s) = 0$.

Where u is the input function, and s is the solution to an unknown partial differential equation (and also a function).

Our PDE solution operator will be $G(u) = s$.

We can express the general solution of our PDE as the operator $G(u)(y) = s(y)$.

A common instantiation is the approximation of the second order elliptic PDE:

$$-\nabla \cdot (a(x)\nabla u(x)) = f(x), \qquad x \in D \tag{1}$$

$$u(x) = 0, \qquad x \in \partial D \tag{2}$$

Where u is our target solution object, $f(x)$ is the right end term, which is regarded as a forcing term in many physical equations, and this $a(x)$ is the parameter, $a \in \mathcal{A}$, where $\mathcal{A}$ is the parameter space. If we write it in the form of the operator, we express it in the following form

$$(L_\alpha u)(x) = f(x), x \in D \tag{3}$$

$$u(x) = 0, x \in \partial D \tag{4}$$

Where $\mathcal{L}_a \cdot = -\operatorname{div}(a\nabla\cdot)$, the subscript a means the operator $\mathcal{L}$ at the parameter a.

Under rather general conditions of $\mathcal{L}_a$, we can define the Green function $G : D \times D \to \mathbb{R}$ as the unique solution to the problem:

$$\mathcal{L}_a G(\cdot, x) = \delta_x \tag{5}$$

To be concrete we will consider infinite-dimensional spaces which are Banach spaces of real-valued functions defined on a bounded open set in $\mathbb{R}^d$. We then consider mappings $\mathcal{F}^\dagger$ which take input functions to a PDE and map them to solutions of the PDE, both input and solutions being real-valued functions on $\mathbb{R}^d$.

 doi:10.20944/preprints202502.1895.v1

2.2. PDE Setting

Let $\mathcal{A}$ and U be separable Banach spaces and $\mathcal{F}^\dagger : \mathcal{A} \to U$ a (typically) non-linear map.

For some bounded, open domain $D \in \mathbb{R}$ and a fixed source function f. For a given function a, the equation has a unique weak solution u, so we can define the solution operator $\mathcal{F}^\dagger$ as the map $a \to u$ of the function to the function.

Our goal is to learn an operator $\mathcal{F}$ to approximate $\mathcal{F}^\dagger$ by using a limited set of observed input-output pairs $\{a_j, u_j\}_{j=1}^N$, where each a_j and u_j are a function on D.

We aim to build an approximation of $\mathcal{F}^\dagger$ by constructing a parametric map

$$\mathcal{F} : \mathcal{A} \times \theta \to U \tag{6}$$

for some finite-dimensional parameter space θ and then choosing $\theta^\dagger \in \Theta$ so that $\mathcal{F}(\cdot, \theta^\dagger) \approx \mathcal{F}^\dagger$.

So our optimization problem goal is the minimization problem of the loss function

$$\min_{\theta \in R^p} E_{a\sim u} ||\mathcal{F}^\dagger(a) - \mathcal{F}_\theta(a)||_u^2 \approx \min_{\theta \in R^p} \frac{1}{N} \sum_{i=1}^{N} ||u^{(i)} - \mathcal{F}_\theta(a^{(i)})||_u^2 \tag{7}$$

which directly parallels the classical finite-dimensional setting.

This is an example of a quote.

3. Algorithm

3.1. Setting of Input Signal and Function Space

The input signal $u(x)$ serves as the key factor for a model to gather characteristic information, and its quality sets the upper limit of the model's achievable accuracy. In this paper, we primarily consider two function spaces:

The Orthogonal (Chebyshev) Polynomial Component of the Gaussian Random Field (GRF) and the Gaussian Random Field (GRF).

Typically, we employ a zero - mean Gaussian random field, expressed as:

$$u \sim \mathcal{G}(0, k_l(x_1, x_2)) \tag{8}$$

where the covariance function is the Gaussian kernel given by:

$$k_l(x_1, x_2) = exp(-||x_1 - x_2||^2 / 2l^2) \tag{9}$$

The length scale l here is of great significance as it determines the smoothness of the sampling function. When the value of l is large, the resulting u function will be smoother, which is of great importance for studying phenomena at different scales.

After selecting the function space, sampling needs to be carried out from this space. The specific approach is to scatter points within the specified region to obtain sample points. Subsequently, the $(\mathbf{u}, \mathbf{x}, \mathbf{y})$ is integrated, and the resulting value is used as the input data of the model. By constructing the model in this way, an approximate solution y is obtained. To comprehensively evaluate the performance of the model, multiple different test sets u are generated by randomly selecting data, so as to examine the performance of the model under different data distributions.

Gaussian random field with the radial-basis function kernel:

Suppose there is a stochastic process $X(t)$ that follows a specific Gaussian random field distribution, that is:$X(t) \sim \mathcal{G}(0, \exp(-\frac{|x-y|^2}{l^2}))$.

 doi:10.20944/preprints202502.1895.v1

Then,$X(t)$ can be represented by the following integral from:

$$X(t) = \sqrt{2}(\pi)^{\frac{1}{4}} \int_{\mathbb{R}^+} (l)^{\frac{1}{2}} \cos(\omega t) \exp(-\frac{l^2\omega^2}{8}) dW(\omega) \quad -\sqrt{2}(\pi)^{\frac{1}{4}} \int_{\mathbb{R}^+} (l)^{\frac{1}{2}} \sin(\omega t) \exp(-\frac{l^2\omega^2}{8}) dB(\omega), \tag{10}$$

In the above expression, W and B are independent standard Brownian motions, which introduce randomness to $X(t)$. To further simplify the expression, a variable substitution is carried out, apply the change of variable $\lambda = l\omega$ and. After the transformation, $X(t)$ can be rewritten as:

$$X(t) = \sqrt{2}(\pi)^{\frac{1}{4}} \int_{\mathbb{R}^+} \cos(\frac{\lambda}{l} t) \exp(-\frac{\lambda^2}{8}) dW(\lambda) - \sqrt{2}(\pi)^{\frac{1}{4}} \int_{\mathbb{R}^+} \sin(\frac{\lambda}{l} t) \exp(-\frac{\lambda^2}{8}) dB(\lambda). \tag{11}$$

Applying a linear interpolation Π_1 on the interval $[t_i, t_{i+1}]$, then

$$\begin{aligned} \mathbb{E}[(X(t) - \Pi_1 X(t))^2] =& \ 2(\pi)^{\frac{1}{2}} \int_{\mathbb{R}^+} ((I - \Pi_1)\cos(\frac{\lambda}{l} t))^2 \exp(-\frac{\lambda^2}{4}) d\lambda \\ &+ 2(\pi)^{\frac{1}{2}} \int_{\mathbb{R}^+} ((I - \Pi_1)\sin(\frac{\lambda}{l} t))^2 \exp(-\frac{\lambda^2}{4}) d\lambda \\ \leq& \ (\pi)^{\frac{1}{2}} (t_{i+1} - t_i)^4 \int_{\mathbb{R}^+} (\frac{\lambda}{l})^4 \exp(-\frac{\lambda^2}{4}) d\lambda \\ =& \ 24\pi \frac{(t_{i+1} - t_i)^4}{l^4}, \end{aligned} \tag{12}$$

where we recalled the error estimate of the linear interpolation on $[a, b]$ (by Taylor's expansion)

$$|(I - \Pi_1)g(t)| = \frac{1}{2}(b - a)^2 |f''(\xi)|, \tag{13}$$

where ξ lies in between a and b. Then by the Borel-Cantelli lemma, we have

$$|X(t) - \Pi_1 X(t)| \leq C \frac{(t_{i+1} - t_i)^{2-\epsilon}}{l^2}, \quad \epsilon > 0, \tag{14}$$

where C is an absolute value of a Gaussian random variable with a finite variance. Therefore, taking a piecewise linear interpolation of $X(t)$ with m points will lead to convergence with order $\mathcal{O}(\frac{1}{m^2 l^2})$.

3.2. Structure Based on U - Net Neural Network and Fourier Neural Operator

The U-Net neural network, as a classic deep-learning architecture, has a unique structure. The first half is mainly responsible for feature extraction, while the second half focuses on the up-sampling operation. This structure is called the encoder-decoder structure. In this study, a Fourier neural operator with adjustable parameters is introduced. By skillfully integrating the feature information extracted by the U-Net neural network into the Fourier operator, the operational efficiency and solution accuracy of the algorithm can be significantly improved.

Step1: When constructing the algorithm framework, P and Q are defined to represent the left - contracting path and the right-expanding path of the U-Net neural network respectively:

$$\begin{aligned} &P : \mathbb{R}^{d_i} \times \Theta \to \mathbb{R}^{d_f} \\ &(\mathcal{P})(x, \theta_R) = P(u(x), \theta_R) \end{aligned} \tag{15}$$

Among them, d_f represents the number of channels used in the neural network. Usually, the value of d_f is greater than d_i or d_0. In the actual operation, the left - contracting path P of the U-Net is used to

 doi:10.20944/preprints202502.1895.v1

lift the input data to a high-dimensional channel space, and then the right - expanding path Q is used to project the data in the channel space to the final output space.

The left encoder network P has a structure similar to that of a convolutional neural network. For a given input function a: $a : \mathbb{D} \in \mathbb{R}^{dA}$, its operation process includes two repeated 3×3 convolution operations, a rectified linear unit (ReLU activation function), and a 2×2 max-pooling layer. When performing the down-sampling operation, the algorithm step size is set to 2.

Convolutional layer:

$$\mathrm{Conv}(a, W, b) = a * W + b \tag{16}$$

. Where $*$ is convolution operation, W is convolution kernel and b is bias.

ReLU activation functions:

$$\mathrm{ReLU}(z) = \max(0, z) \tag{17}$$

Pooling layer: The maximum pooling layer is usually used to reduce the spatial dimension and help to obtain. Enter the features of the image:

$$\mathrm{MaxPool}(a) = \max_{\text{window}}(a) \tag{18}$$

Input a into the encoder (left-contracting path P) of the U-net neural network. Through convolution, ReLU activation function, and max-pooling layer operations, the input is lifted to the high dimensional channel space $\mathbb{R}^{d_f}$.

Step2: Then we introduce the nonlinear integral Fourier operator $(\kappa(\phi)v)(x) = \mathcal{F}^{-1}(R_t \cdot \mathcal{F}v_t)(x)$

$$\mathbf{F}_i : \{v_t : D_t \to \mathbb{R}^{d_{v_t}}\} \to \{v_{t+1} : D_{t+1} \to \mathbb{R}^{d_{v_{t+1}}}\}, t \in \{0, 1, 2, ..., T\} \tag{19}$$

Apply four-layer integral operators and activation functions to the obtained feature v_t:, and update it specifically according to the formula

$$v_{t+1}(x) = \sigma(Wv_t(x) + \mathcal{F}^{-1}(R_t(k) \cdot \mathcal{F}v_t)(x)) \tag{20}$$

Where σ is an activation function $GELU$, $R_t : \mathbb{Z}^d \ to \mathbb{C}^{d_{t+1} \times d_t}$ is a linear transformation. It can handle lower Fourier patterns and filter out higher patterns, $W \ in \mathbb{R}^{to?n \times n}$ and ϕ into the nucleus $\kappa_\phi : \mathbb{R}^{2(d+1)}$ to $\mathbb{R}^{n \times n}$, this is all something to learn from the data. Step3: The resulting result v_t is again input into the U-Net neural network decoder Q, deconvolved, jump connections, and other operations, which help to retain high-resolution details.

$$\begin{aligned} &Q : \mathbb{R}^{d_f} \times \Theta \to \mathbb{R}^{d_o} \\ &(\mathcal{Q})(x, \theta_Q) = Q(u(x), \theta_Q) \end{aligned} \tag{21}$$

Similar to the encoder, the decoder also applies the convolution and activation functions to handle the feature graph.

Input the updated result into the decoder (right-expanding path Q) of the U-net neural network. Process it through operations such as de-convolution and skip connection, and finally output $u(x)$.

Step4: The final output results and the original sample to calculate the loss through the loss function, and finally through the loss function to optimize the U-Net network and the Fourier operator to obtain the optimal parameter value.

$$L = MSE_F + \lambda MSE_{bc/initial}. \tag{22}$$

 doi:10.20944/preprints202502.1895.v1

Where

$$MSE_{bc/initial} = \frac{1}{N_f}\sum_{N_f}^{i=1}(\mu(t_0^i, x_0^i) - \mu_0^i)^2 and MSE_F \qquad = \frac{1}{N_f}\sum_{N_f}^{i=1}(\mathcal{F}^{\dagger}(a_0^i) - \mathcal{F}_\theta(a_0^i)) \tag{23}$$

Here, $\{t_0^i, x_0^i, a_0^i, u_0^i\}_{i=1}^N$ represents the initial and boundary training datasets. The loss term $MSE_{bc/initial}$ corresponds to constraints imposed by the initial data and boundary data.

$$L = MSE_{\mathbf{F}} + \lambda' MSE_f \tag{24}$$

$$MSE_f = \frac{1}{N_u}\sum_{i=1}^{N_u}|f(t_f^i, x_f^i)|^2 \text{ and } MSE_{\mathbf{F}} = \frac{1}{N_{\mathbf{F}}}\sum_{i=1}^{N_{\mathbf{F}}}|\mathbf{F}^{\dagger}(a^i) - \mathbf{F}_\theta(a^i)|. \tag{25}$$

Here, the set $\{t_f^i, x_f^i\}_{i=1}^N$ represents the collocation points for $f(t,x)$, with the loss function MSE_f enforcing the structural constraints imposed by equations.

Optimize the parameters of the U-Net network and the Fourier operator through the loss function to obtain the best parameter values and minimize the loss.

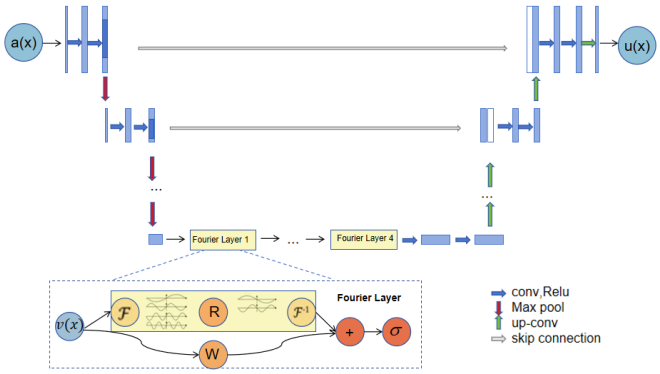


Figure 1. The architecture of the neural operators.

(a) The complete structure of the neural operator: The entire operation process of the neural operator starts from the input a. First, the input data is upgraded to a high-dimensional channel space through the U-Net encoder. Then, four layers of integral operators and activation functions are applied to deeply process the data. After that, the processed data is sent to the target dimension through the U-Net decoder, and finally, u is output.

(b) Fourier layer: The operation of the Fourier layer starts from the input ν. First, the Fourier transform $\mathcal{F}$ is applied to transform the data into the frequency domain. Then, a linear transformation R is performed on the low-frequency signals to filter out the high-frequency signals. After that, the inverse Fourier transform $\mathcal{F}^{-1}$ is applied to transform the data back into the time domain. At the bottom, a local linear transformation W is also applied to further process the data. $\nu_{t+1}(x) = \sigma(W\nu_t(x) + (\kappa(a;\phi)\nu_t)(x)$

The above formula can be seen as a combination of a linear connection and a nonlinear transformation, and then through the activation function.

The nonlinear transformation κ is implemented through a kernel integration operator: $(\kappa(a;\phi)\nu_t)(x) = \int_D \kappa(x,y,a(x),a(y);\phi)\nu_t(y)dy$

The choice kernel form is $\kappa_\phi(x,y,a(x),a(y)) = \kappa_\phi(x-y)$, the above operation is formally similar to the convolution operation, so it can be expressed by the Fourier transform: $(\kappa(a;\phi)\nu_t)(x) = \mathcal{F}^{-1}(\mathcal{F}(\kappa_\phi\cdot)\mathcal{F}(\nu_t))(x)$

 doi:10.20944/preprints202502.1895.v1

A parameter matrix R_ϕ is introduced, which can transform the lower Fourier patterns and filter out the higher patterns.The Fourier neural operator is finally expressed as follows: $(\kappa(a;\phi)v_t)(x) = \mathcal{F}^{-1}(R_\phi \cdot)\mathcal{F}(v_t))(x)$

Where $\kappa \in D$, $\mathcal{F}v_t(\kappa) \in C^{d_v}$, $R_\phi(\kappa) \in C^{d_v \times d_v}$ is both of A function space.

4. Experiment

This section mainly uses the UNO model to solve the 1-d Burgers' equation, the 2-d Darcy Flow equation and the 2-d Navier-Stokes equation. First, we introduce the experimental configuration and the experimental parameter settings. Then explain the experimental effect section. Finally, solve the partial differential equation according to the model framework introduced and proposed in this chapter, comparing the merits of this model for solving partial differential equations, we get the solution speed, the solution result, and the error with the real solution of the model in the three equations.

4.1. Burgers

In this paper, we choose the 1-d Burgers equation, which is a nonlinear partial differential equation. The equations considered in this paper are as follows, formally:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \mu\frac{\partial^2 u}{\partial x^2}, x \in (0,1), t \in (0,1],$$

$$u(x, t = 0) = u_0(x), 0 < x < 1.$$

Where $u(x,t)$ is the unknown function, which depends on the time variable t and the spatial variable x, $\mu \in \mathbb{R}_+$ is the viscosity coefficient, the initial condition is $u_0(x) \in L^2_{per}((0,1);\mathbb{R})$, and the boundary condition is the periodic boundary condition.

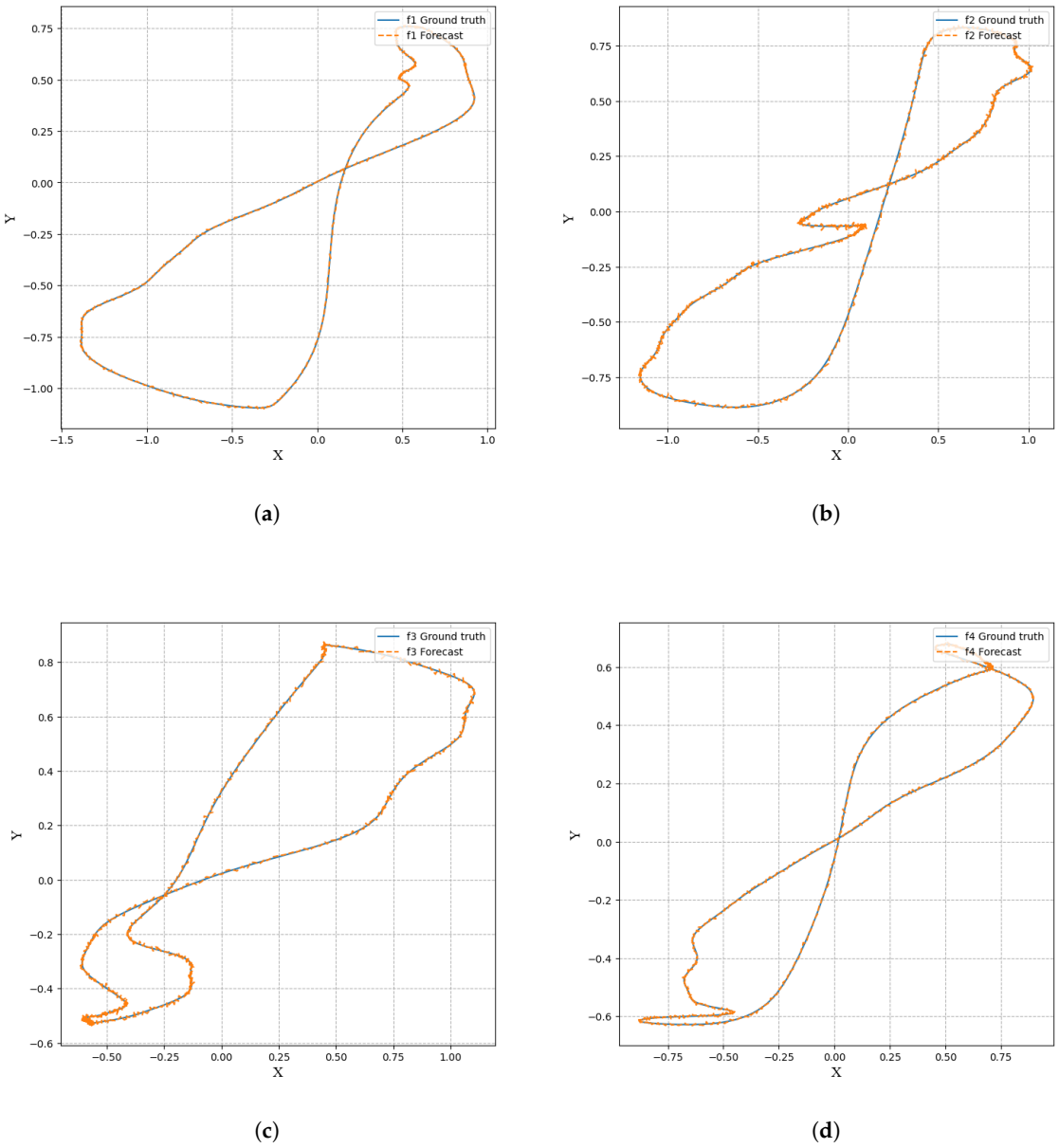
According to $u_0(x)$ randomly sampling $u_0 \sim \mu, \mu = \mathcal{N}(0, 625(-\Delta + 25I)^{-2})$, modes = 16, width = 64, batch size=20. Forced term =0, viscosity coefficient $\mu = 0.1$, grid $2^{13} = 8192$, other grids directly subsampled. Network input $a(x) = u(x, t = 0) = u_0(x)$, output $u(x, t = 1)$, we have 8192 $a(x)$ in 2048 grid samples, here is the average sampling, neural network input dimension (20,128,2), input neural operator dimension after dimension (20,128,64), the accuracy of the equation label determines the model accuracy, if the label accuracy is poor, the model has no sense.

The approximate solution is u, and the true solution is uacc, with an error of $O(h^2)$, $u - u_{acc} = O(h^2)$. The solver is GPU, the error is $G(f,x,y) - u(x,y) = O(h^p)$, G training label is approximate solution.

$$G(f,x,y) - u_{acc}(x,y) = G(f,x,y) - u(x,y) + u(x,y) - u_{acc}(x,y) = G(f,x,y) - u_{acc}(x,y) = O(h^2)$$

It is expected that any f=net (x,y) has G (f, x, y) satisfying the solution constraint regularization of the corresponding PDE.

 doi:10.20944/preprints202502.1895.v1



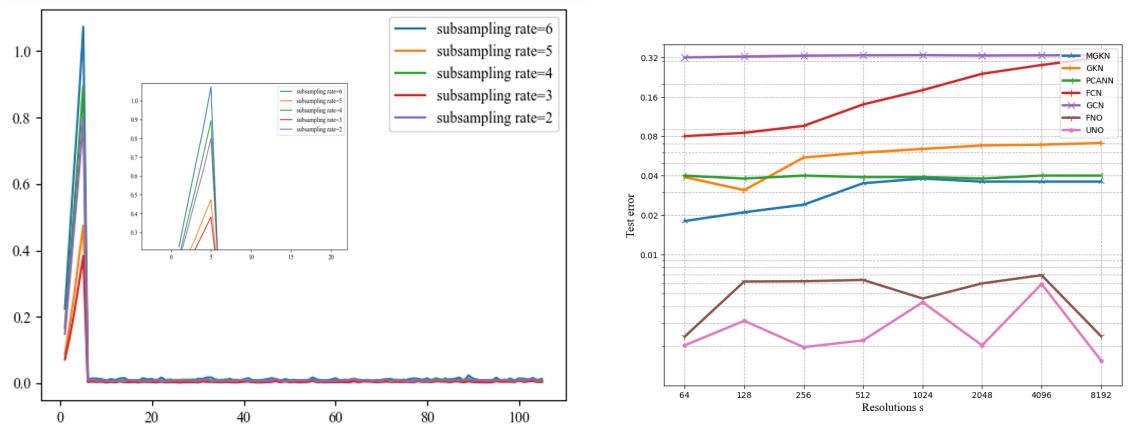
(**a**) (**b**)

(**c**) (**d**)

Figure 2. (**a**) 100 solutions visulization.(**b**) 1000 solutions visulization. (**c**) 2000 solutions visulization. (**d**) 8000 solutions visulization.

Figure 2 randomly intercepts several algorithm results from different periods, showing the error between the prediction and the true value of the algorithm, without much change to the naked eye, and the visualization effect is good.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 February 2025 doi:10.20944/preprints202502.1895.v1



(**a**) (**b**)

Figure 3. (**a**) Epochs(**b**) Algorithm contrast.

Table 1. Experiment 1 results.

Network	s=64	s=128	s=256	s=512	s=1024	s=2048	s=4096	s=8192
MGKN	0.0187	0.0223	0.0243	0.0355	0.0374	0.0360	0.0364	0.0364
GKN	0.0392	0.0325	0.0573	0.0614	0.0644	0.0687	0.0693	0.0714
PCANN	0.042	0.0391	0.0398	0.0395	0.0391	0.0383	0.0392	0.0393
FCN	0.0827	0.0891	0.1158	0.1407	0.1877	0.2313	0.2855	0.3238
GCN	0.3211	0.3216	0.3359	0.3438	0.3476	0.3457	0.3491	0.3498
FNO	0.0024	0.0063	0.0063	0.0066	0.0048	0.0060	0.0069	0.0024
UNO	0.0021	0.0032	0.0020	0.0023	0.0045	0.0021	0.0060	0.0018

Figure 3 compares the errors as the number of iterations increases when the subsampling rate varies. Moreover, we can see that when the subsampling rate is 3 and 5, the error is minimized and the algorithm is better than other cases. It can be seen from the algorithm comparison diagram on the right of Figure 3 that the different algorithms exhibit the ability to solve the partial differential equations in the one-dimensional Burgs equation experiment. The UNO model uses convolution to extract physical feature information. The jump connection reduces the information loss of the model. The results are more accurate and smaller than other algorithms, which is better than other algorithms.

4.2. Darcy Flow

In this paper, we choose the steady-state 2-d Darcy flow equation for a second-order linear elliptic partial differential equation.

$$-\nabla \cdot (a(x))\nabla u(x) = f(x), x \in (0,1)^2$$

$$u(x) = 0, x \in \partial(0,1)^2$$

Where the boundary condition is the Dirichlet boundary condition, a is the diffusion coefficient, $a(x) \in L^{\infty}((0,1)^2;\mathbb{R})$, $u \in H_0^1((0,1)^2;\mathbb{R})$ is the solution, and the external force $f(x) \in H^{-1}((0,1)^2;\mathbb{R})$.

We processed the dataset, first $\mu = \psi_{\#}\mathcal{N}(0, 625(-\Delta + 9I)^{-2}))$ is a probability measure with zero Neumann boundary conditions on LaPraaca, where $\forall x \geq 0, \psi(x) = 12 and \forall x < 0, \psi(x) = 3$. The external force is fixed to be $f(x) = 1$. The coefficient $a(x)$ was sampled from μ, and the solution u was obtained using the traditional finite element method on the 64×64 and 256×256 grids in Matlab. Datasets with different resolutions can be generated by downsampling. We parameterize a

 doi:10.20944/preprints202502.1895.v1

two-dimensional Fourier neural operator consisting of four Fourier layers, with 20 frequencies per channel, width = 64.

The loss function can be written in the following form:

$$MSE_f = -\nabla \cdot (a(x))\nabla u(x) - f(x)$$

In the experimental section, the FNO model, ResNet model, TFNET model, and UNO model were used to solve the 2-d Darcy flow equation.

The true and model solutions of the equations and their errors are shown in Figuer:

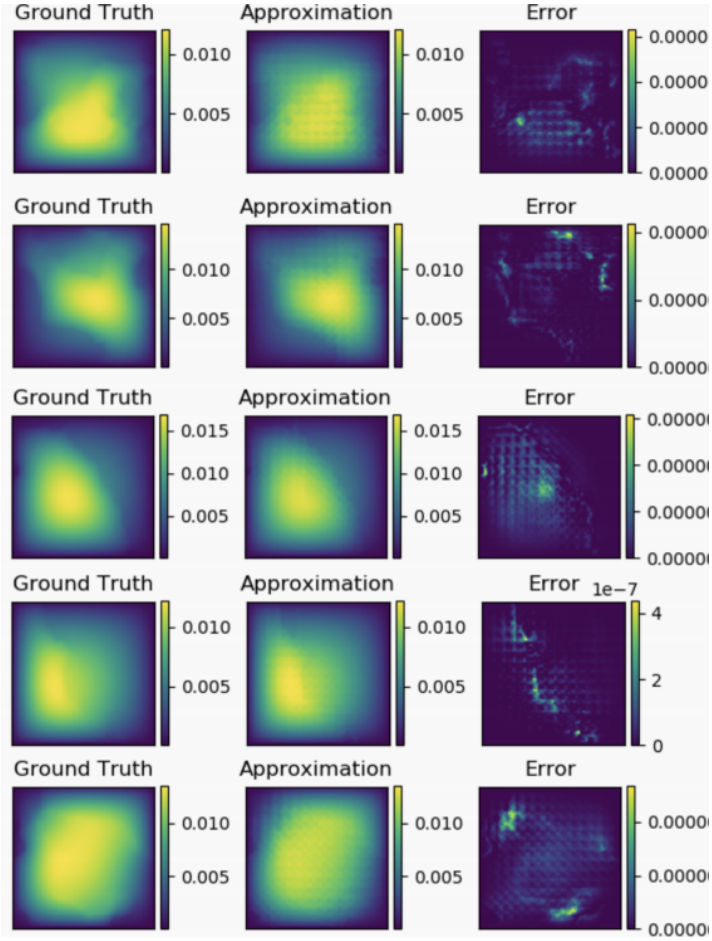
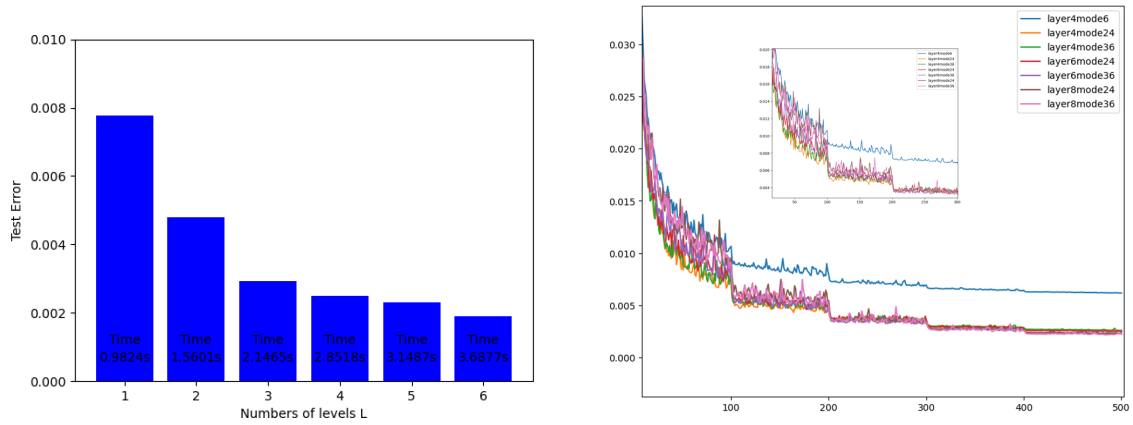


Figure 4. Visualization error map

Figure 4 shows the contrast between the true and predicted values from 1s to 5s and the error situation, which can be seen per second.

 doi:10.20944/preprints202502.1895.v1



(**a**) (**b**)

Figure 5. (**a**) Fourier layer number analysis Fig.(**b**) Visualization error map.

Figure 5 left figure shows that the number of Fourier neural operator layers is different. When the number of Fourier neural operator layers is 1, it takes the least time and the test error is the largest. When the number of levels increases, the time-consuming number increases, but the test error becomes smaller. According to the need, three or four and five layers are the most reasonable, depending on the situation, here take four layer Fourier operator. The right image introduces the effect of the different number of Fourier neural operator layers and the filtered models on the algorithm error as the number of iterations increases. As the number of iterations increases, the error becomes smaller. When the four-layer Fourier neural operator and the filtered out movies are 24, the effect is better and the performance is better. Then the algorithm model selects the four-layer Fourier neural operator.

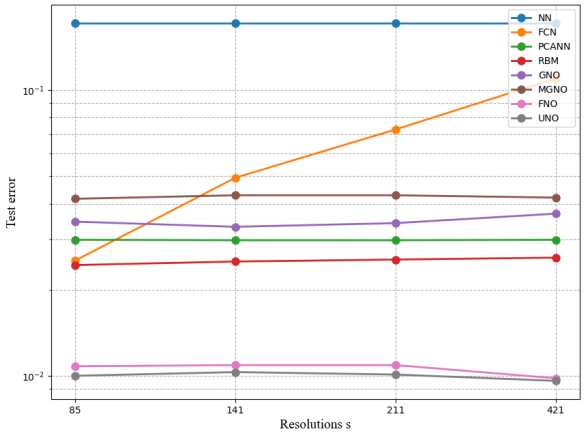


Figure 6. The training error plots for each model.

 doi:10.20944/preprints202502.1895.v1

Table 2. Experiment 2 results.

Network	s=85	s=141	s=211	s=421
NN	0.1716	0.1716	0.1716	0.1716
FCN	0.0253	0.0493	0.0727	0.1097
PCANN	0.0299	0.0298	0.0298	0.0299
RBM	0.0244	0.0251	0.0255	0.0259
GNO	0.0346	0.0332	0.0342	0.0369
MGNO	0.0416	0.0428	0.0428	0.0420
FNO	0.0122	0.0124	0.0125	0.0099
UNO	0.0108	0.0109	0.0109	0.0098

It can be seen from Figure 6 that when different algorithms contrast error maps at resolutions of 85×85, 141×141, 211×211, 421×421, we can see that UNO has lower and more stable error loss values. Compared than the most similar FNO algorithm.

4.3. Navier-Stokes

In this paper, the incompressible fluid equation of motion namely the two-dimensional Navier-Stokes equation is chosen for experiments. The equation form used in this paper is as follows:

$$\partial_t w(x,t) + u(x,t)\nabla w(x,t) = \nu \nabla w(x,t) + f(x), x \in (0,1)^2, t \in [0,T]$$

$$\nabla u(x,t) = 0, x \in (0,1)^2, t \in [0,T]$$

$$w(x,t=0) = w_0(x), x \in \Omega.$$

Where $u(x,t)$ shows the fluid velocity, ∇ shows the gradient operator, $u_0(x) \in L^2_{per}((0,1);\mathbb{R})$represents the initial condition, namely, the initial vorticity, $\nu \in \mathbb{R}_+$ represents the viscosity coefficient,and $f(x) \in L^2_{per}((0,1);\mathbb{R})$ for the external force acting on the fluid. This article sets the external force to$f(x) = 0.1(sin(2\pi(x_1+x_2)) + cos(\pi(x_1+x_2)))$.

The experimental scenario is the vorticity form of the sticky non-compresdable NS equation in 2 dimensions. The vorticity field is simplified by directing on x and y to obtain the velocity field u and v. Using the NS equation. Using the vorticity equation as the loss function, the output layer of the neural network needs a neuron to represent w, and the automatic differentiation only needs to guide this one variable, which is often less difficult to optimize. This scenario inputs the vorticity field variable at time t at [0,10) to the FNO model, with a resolution of 64×64 and 256×256, that is, to generate the dataset, the initial vorticity condition w_0 was sampled randomly from the distribution $w_0 \sim \mu, \mu = \mathcal{N}(0, 7^{3/2}(-\Delta + 49I)^{-2.5})$, with periodic boundary conditions. The input variable dimension is [64,64,10]and[256,256,10]. It is hoped that the FNO model can output the value of the vorticity field at time $t > 10$, that is, learn the evolution relationship of the vorticity time series, and realize a model similar to the time series prediction.

The loss functions are defined as follows:

$$MSE_f = \partial_t w(x,t) + u(x,t)\nabla w(x,t) - \nu \nabla w(x,t) + f(x)$$

 doi:10.20944/preprints202502.1895.v1

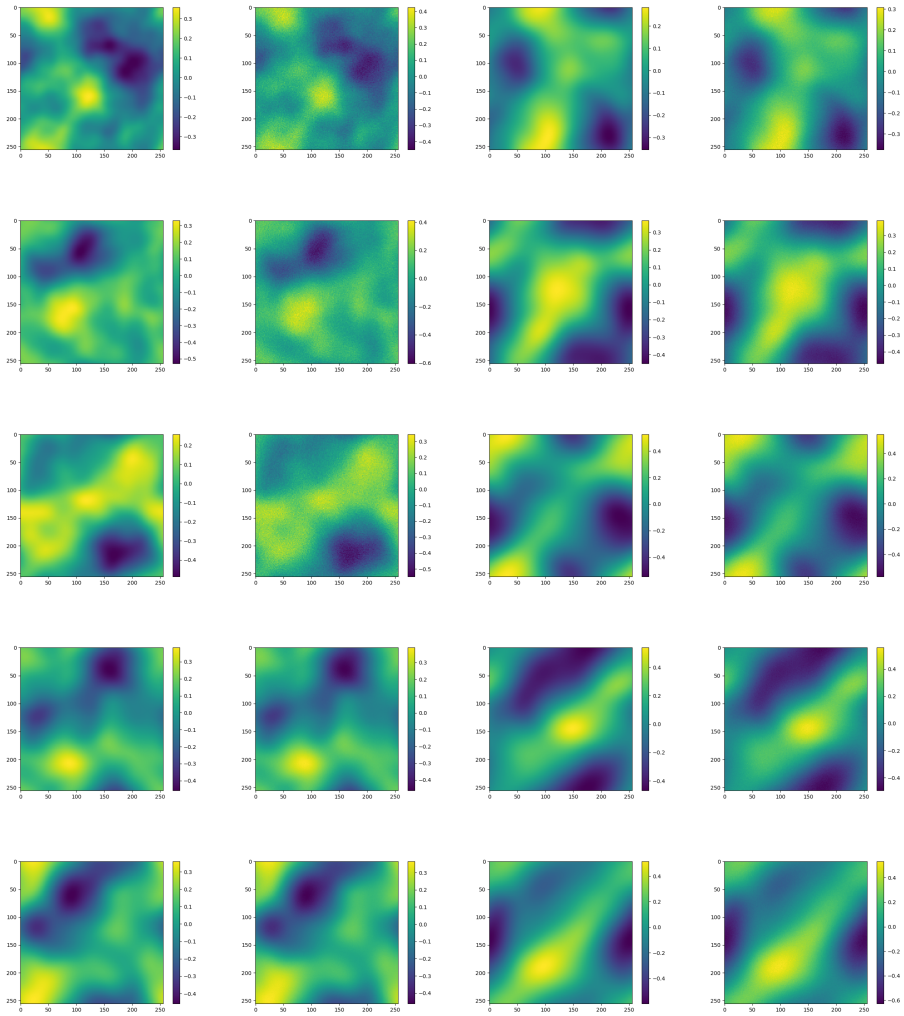
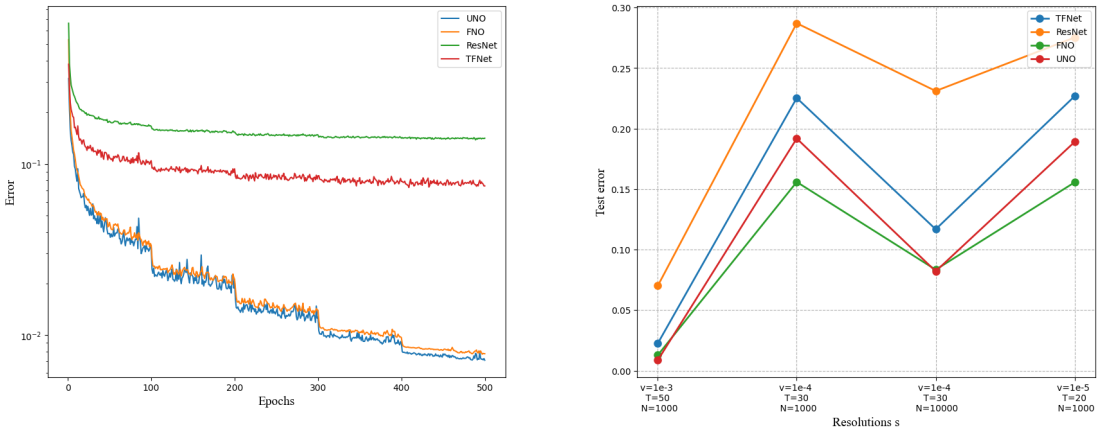


Figure 7. Visualization error map

Figure 7 shows the images of true and predicted values from 1 second to 10 seconds respectively from top to bottom. As can be seen from the image comparison per second, the visualization results are relatively consistent with small error.



(**a**) (**b**)

Figure 8. (**a**) Epochs.(**b**)Resolutions.

 doi:10.20944/preprints202502.1895.v1

Table 3. Experiment 3 results

Network	total number of parameters	each round of training takes time	v=1e-3 T=50 N=1000	v=1e-4 T=30 N=1000	v=1e-4 T=30 N=10000	v=1e-5 T=20 N=1000
UNO	414,517	38.99s	0.0128	0.1879	0.0834	0.1856
FNO	6,558,53	45.80s	0.0135	0.1551	0.0835	0.1524
ResNet	266,641	78.47s	0.1716	0.2871	0.2311	0.2753
TF-Net	7,451,724	47.21s	0.0225	0.2253	0.1168	0.2268

From Figure 8, FNO and UNO are found to have better results. By comparing time and data points,suitable algorithms are selected. The error of UNO and FNO algorithm can continue to decrease steadily with the number of iterations, which has great advantages compared with other models. The error of UNO algorithm drops faster, with an obvious decrease process at 100, 200, 300 and 400, showing a continuous jitter decreasing trend, and has obvious advantages in solving the equation. The RENET and TFNET models are less effective, and compared with UNO and FNO, the curves are relatively smooth and the errors are larger during the iteration. Figure 8 also compares the experimental error of UNO and FNO, RENET, and TFNET on the dataset. The figure shows the data points N=10000 and T=50. When the viscosity coefficient is $1e-3$, the UNO error is the smallest, and UNO has higher accuracy. When the data point N=1000, UNO is worse than FNO, but when the data pont increases, N=10000, the UNO error is always the smallest compared with other algorithm models.

5. Conclusions

This paper proposes an innovative method based on the combination of U-Net neural networks and Fourier neural operators, aiming to improve the accuracy and efficiency of solving partial differential equations (PDEs). By integrating the unique encoder-decoder structure and skip connections of U-Net with the deep feature processing capabilities of Fourier transforms in the frequency domain, the method fully leverages the strengths of U-Net in spatial feature extraction and the capabilities of Fourier transforms in frequency-domain feature processing. This combination not only preserves spatial details but also captures both high-frequency and low-frequency characteristics of PDE solutions through frequency-domain analysis, thereby significantly enhancing solving accuracy. Experimental results demonstrate that, compared to traditional methods, this approach performs exceptionally well in various complex PDE-solving tasks, achieving higher accuracy. While the method reduces reliance on large-scale datasets, high-quality datasets remain crucial for training the model when dealing with complex physical systems. Future research could explore ways to further improve the model's generalization ability in data-scarce scenarios.

Author Contributions: - TZ: Conceptualized and designed the research plan, took the lead in the collection and preliminary analysis of experimental data. Additionally, TZ polished the language and proofread the format of the entire paper, ensuring its professionalism in expression and presentation. - XRZ: Conducted in - depth statistical analysis of the data, provided key insights into the interpretation of the results, and participated in writing the core content of the discussion section, making significant contributions to the discussion and interpretation of the research findings

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

 doi:10.20944/preprints202502.1895.v1

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Weinan. E, Machine learning and computational mathematics, arXiv preprint. arXiv:2009.14596.(2020)
2. Han, Jiequn, and Arnulf Jentzen, Deep learning-based numerical methods for high-dimensional parabolic par- tial differential equations and backward stochastic differential equations, Communications in mathematics and statistics. 5.4 (2017): 349-380.
3. Weinan. E, Jiequn Han, and Arnulf Jentzen, Algorithms for solving high dimensional PDEs: from nonlinear Monte Carlo to machine learning, Nonlinearity. 35.1 (2021): 278.
4. Weinan E and Bing Yu, The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems, Communications in Mathematics and Statistics. 6(1):1?12, 2018
5. Turner, M. Jon, et al, Stiffness and deflection analysis of complex structures, Journal of the Aeronautical Sciences. 23.9 (1956): 805-823.
6. Richtmyer, Robert D., and Keith W. Morton, Difference methods for initial-value problems, Malabar. (1994).
7. Jameson, Antony, Wolfgang Schmidt, and Eli Turkel, Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes, 14th Fluid and Plasma Dynamics Conference. (1981).
8. Dockhorn T. A discussion on solving partial differential equations using neural networks[J]. arXiv preprint arXiv:1904.07200, 2019.
9. Hornik K., Stinchcombe M., White H. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks[J]. Neural Networks, 1990, 3(5): 551-560.
10. Lagaris I., Likas A., Fotiadis D. Artificial neural networks for solving ordinary and partial dif ferential equations[J]. IEEE Transactions on Neural Networks, 1998, 9(5): 987-1000.
11. Lagaris I., Likas A., Fotiadis D. Artificial neural networks for solving ordinary and partial dif ferential equations[J]. IEEE Transactions on Neural Networks, 1998, 9(5): 987-1000.
12. Winovich N., Ramani K., Lin G. Convpde-uq: Convolutional neural networks with quantified uncertainty for heterogeneous elliptic partial differential equations on varied domains[J]. Journal of Computational Physics, 2019, 394: 263-279
13. Long Z.C., Lu Y.P., Ma X.Z., Dong B. Proceedings of machine learning research: volume 80 PDE-net: Learning PDEs from data[M]. PMLR, 2018: 3208-3216.
14. Long Z.C., Lu Y.P., Dong B. Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network[J]. Journal of Computational Physics, 2019, 399: 108925.
15. LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.
16. Fan, Yuwei, et al, A multiscale neural network based on hierarchical matrices, Multiscale Modeling Simulation. 17.4 (2019): 1189-1213.
17. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators[J]. Neural networks, 1989, 2(5): 359-366.
18. Mathieu M, Henaff M, LeCun Y. Fast training of convolutional networks through ffts[J]. arXiv preprint arXiv:1312.5851, 2013.
19. Vahid K A, Prabhu A, Farhadi A, et al. Butterfly transform: An efficient fft based neural architecture design[C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020: 12021-12030.
20. Kovachki, Nikola, et al, Neural operator: Learning maps between function spaces,arXiv preprint. arXiv:2108.08481 (2021).
21. Li, Zongyi, et al, Fourier neural operator for parametric partial differential equations,arXiv preprint. arXiv:2010.08895 (2020).

 doi:10.20944/preprints202502.1895.v1

22. Kushnure D T and Talbar S N.Ms-Unet: A Multi-Scale Unet with Feature Recalibration Approach for Automatic Liver and Tumor Segmentation in Ct Images[J].Computerized Medical Imaging and Graphics,2021,89:101885.
23. Isensee F, Kickingereder P, Wick W,et al.Brain Tumor Segmentation and Radiomics Survival Prediction: Contribution to the Brats 2017 Challenge[C]//Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: Third International Workshop, BrainLes 2017, Held in Conjunction with MICCAI 2017, Quebec City, QC, Canada, September 14, 2017, Revised Selected Papers 3, 2018.Springer,287-297.