IGA-Reuse-NET

A deep-learning-based isogeometric analysis-reuse approach with topology-consistent parameterization[Formula presented]

Wang, Dandan; Xu, Jinlan; Gao, Fei; Wang, Charlie C.L.; Gu, Renshu; Lin, Fei; Rabczuk, Timon; Xu, Gang

DOI
10.1016/j.cagd.2022.102087

Publication date
2022

Document Version
Final published version

Published in
Computer Aided Geometric Design

Computer Aided Geometric Design 95 (2022) 102087

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

IGA-Reuse-NET: A deep-learning-based isogeometric analysis-reuse approach with topology-consistent parameterization

Dandan Wang [a], Jinlan Xu [a], Fei Gao [a], Charlie C.L. Wang [b,d], Renshu Gu [a], Fei Lin [a], Timon Rabczuk [c], Gang Xu [a,*]

[a] School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China
[b] The University of Manchester, United Kingdom
[c] Institute of Structural Mechanics, Bauhaus-University Weimar, Marienstr. 15, D-99423 Weimar, Germany
[d] Delft University of Technology, the Netherlands

ARTICLE INFO

Article history:
Available online 15 April 2022

Keywords:
Isogeometric analysis
Deep learning
Topology-consistent model
Analysis-reuse
CAD/CAE integration

ABSTRACT

In this paper, a deep learning framework combined with *isogeometric analysis* (IGA for short) called IGA-Reuse-Net is proposed for efficient reuse of numerical simulation on a set of topology-consistent models. Compared with previous data-driven numerical simulation methods only for simple computational domains, our method can predict high-accuracy PDE solutions over topology-consistent geometries with complex boundaries. UNet3+ architecture with *interlaced sparse self-attention* (ISSA) module is used to enhance the performance of the network. In addition, we propose a new loss function that combines a coefficients loss and a numerical solution loss. Several training datasets with topology-consistent models are constructed for the proposed framework. To verify the effectiveness of our approach, two different types of Poisson equations with different source functions are solved on three datasets with different topologies. Our framework can achieve a good trade-off between accuracy and efficiency. It outperforms the *physics-informed neural network* (PINN for short) model and yields promising results of prediction.

1. Introduction

Isogeometric analysis (IGA for short) method proposed by Hughes et al. [Hughes et al. (2005)], offers a methodology of seamless integration of numerical simulation and geometric modeling. However, computational efficiency is one of the major challenges in the current development of IGA. High-order spline basis functions are often employed to achieve smooth solution fields with high continuity, which however also increases the computational cost in the step of filling stiffness matrices [Xu et al. (2017)]. In this paper, we propose a novel method for *analysis reuse* in IGA on geometries with similar semantic features, by which the computational efficiency can be significantly improved. The physical analysis on a family of products having the same topology but different shapes can benefit from this technique to generate results in a very efficient

The code (and data) in this article has been certified as Reproducible by Code Ocean: https://codeocean.com/. More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail address: gxu@hdu.edu.cn (G. Xu).

https://doi.org/10.1016/j.cagd.2022.102087
0167-8396/

way. More interestingly, this method can be employed in the shape optimization routine as the inner loop to generate a function-driven inverse design.

In recent years, deep learning has developed rapidly and has shown excellent performance in various fields such as image processing, computer vision and natural language processing. The power of *deep neural network* (DNN) lies in its universal approximation theory [Hornik et al. (1989)], which allows DNN to approximate a function with high precision by a set of nonlinear operations. Given its high performance and powerful functionality, deep learning is increasingly being applied to other areas of science and engineering – such as solving *partial differential equations* (PDEs).

In 2005, it was proposed to solve PDEs by combining *Finite Element Method* (FEM) and machine learning [Ramuhalli et al. (2005)], but at this time it is more from the traditional simulation methods to find the law and then apply to the *multi-layer sensor* (MLP), does not take full advantage of neural networks. But with the rapid development of deep learning, and the support of frameworks such as tensorflow [Abadi et al. (2016)] and pytorch [Ketkar (2017)], there have been a number of papers in recent years that effectively use DNN to solve PDEs [Anitescu et al. (2019)], such as PINNs proposed by [Raissi et al. (2019)], using automatic differentials of the tool library to calculate the derivatives of the output of neural networks to inputs. Thus, a loss function based on PDEs is constructed. At the same time, the *Convolutional Neural Network* (CNN), which has made great progress in computer vision, has attracted attention, and people are beginning to use CNN to solve PDEs. [Yao et al. (2019)] proposed FEA-Net, which uses the CNN architecture by treating the source function of PDEs on the rectangular mesh as an image process.

In this paper, a deep learning method combined with IGA is studied to solve the Poisson equation on the topology-consistent B-spline models. There are many advantages to solving PDEs with IGA, one of which is to use high-order basis functions to construct smooth solutions with high continuity, but this will increase computational costs when filling the stiffness matrix. In [Xu et al. (2017)], the authors proposed a framework of computation reuse in IGA on a set of topology-consistent models. With the help of this method, evaluation on the stiffness matrix and imposition of the boundary conditions can be pre-computed and partially reused for models having consistent parameterization, and the computation efficiency similar to classical finite element method can be achieved for IGA on these models. However, this method still needs to solve a large linear system [Xu et al. (2017)], and it is a challenging problem to realize completely analysis-reuse for IGA simulation problems over a large set of CAD models. Therefore, it is necessary to propose a method of analysis-reuse in IGA on topology-consistent models using deep learning. Applications that could benefit from this study include (1) numerical simulation on a series of geometries with different shapes but the same topology, and (2) real-time simulation of geometry with changing boundaries. The main contributions of this paper are summarized as follows:

- A deep learning framework combined with IGA called IGA-Reuse-Net is proposed for efficient reuse of numerical simulation on a set of topology-consistent B-spline planar parameterizations. Compared with previous data-driven numerical simulation methods only for simple computational domains, such as the PINN method, our method can predict more accurate PDE solutions over topology-consistent geometries with complex boundaries.
- UNet3+ architecture with ISSA module is proposed to enhance the performance of the network. A coefficients loss and a numerical solution loss, which are combined as the loss function for network training is proposed.
- Several training datasets with topology-consistent models are also constructed for the proposed framework. The proposed framework can achieve a good trade-off between accuracy and efficiency. It outperforms the widely used PINN model and yields promising prediction results.

The structure of this paper is organized as follows. Section 2 introduces the related work on solving PDEs using deep learning. Section 3 introduces the proposed framework in detail including the generation of dataset, the selection of network architecture, and the design of loss function. The effectiveness and the efficiency of our method have been verified by experimental tests in Section 4. Section 5 concludes our paper and outlines the future work.

2. Related work

There has been a long history of interest in solving PDEs using neural networks, and as early as the 1990s, it is suggested that PDEs could be solved in combination with traditional numerical methods and machine learning [Ramuhalli et al. (2005); Yao et al. (2019); Lagaris et al. (1998)]. They find out the rules from the traditional numerical methods and summarize expressions, and then construct linear solvers by connecting neurons and assigning parameters according to the expressions. As a result, the neural networks used in early work are mostly MLP, and the networks do not need to be trained at all.

With the development of machine learning, the emergence of various activation functions not only enhances the approximation ability of the neural network, but also increases the depth of the network. Turbulence calculation, random PDEs and high-dimensional PDEs are all challenging problems in PDEs. [Ling et al. (2016)] and [Duraisamy et al. (2019)] have done some research on the use of DNN to solve turbulence problems. [Beck et al. (2019)] and [Raissi (2018)] use DNN to solve random PDEs and high-dimensional PDEs.

In addition, there are also some previous works to find a more general way to quickly solve a variety of PDEs. One of the works is PINNs proposed by [Raissi et al. (2019)]. PINNs uses the simplest *full-connected network* (FCN). With the automatic differential of the deep learning framework, PINNs can easily obtain the derivatives of the PDE's solution to the physical coordinates of the geometry, which is used to construct the loss function according to the PDE. This method of constructing

loss functions based on PDE is called a physics-based approach. In addition, [Lu et al. (2021)] implements a Python library – DeepXDE based on PINNs to help us use PINNs easily.

At the same time, the full convolution model, which has made great progress in computer vision, has also received great attention. In addition to the above-mentioned use of FCN to solve PDEs, some works are also beginning to use CNN to solve PDEs. FEA-Net proposed by [Yao et al. (2019)] treats the source function of PDEs on the rectangular mesh as an image processing so that the CNN architecture can be used. They extract repeated calculations from the stiffness matrix as convolution cores, thus transforming the calculation of linear equations in FEM into convolutional operations of images. The iterative process in the numerical method is then implemented through a residual network of multiple convolution blocks. However, this method can only solve PDEs on the regular domain, and can not solve on geometries with complex boundaries.

[Gao et al. (2020)] proposes PhyGeoNet, a new CNN architecture based on physical constraints that solves PDEs on irregular domains and does not require numerical simulation solution as labeled data. PhyGeoNet introduces elliptic coordinate mapping, which implements the transformation from irregular computational domain to regular parameterization domain. And [Li et al. (2020)] uses the encoder-decoder based CNN to predict complex time-dependent behavior of the reaction diffusion system with different conditions. They also map the irregular physical domain to the regular parametric domain through parameterization, and then perform operations such as convolution on the parametric domain. But the network input variables are some settings of the PDEs in these works, and these works can only be trained for one geometry.

[Thuerey et al. (2020)] comes up with the idea of training network in a data-driven way that could predict the solution of *Reynolds-average Navier-Stokes* (RANS) on other airfoils the network had never seen before. After obtaining 1505 different airfoils, they construct a dataset by solving the RANS equation on these airfoils using the FEA library – OpenFOAM. This dataset is then used to train the network model of the UNet architecture. The inputs of the network are the $[0, 1]$ representation of the airfoils and the free-stream velocity, and the outputs of the network are the pressure and velocity field on and around the airfoils.

The above-mentioned traditional numerical methods combined with the methods of solving PDE through deep learning are all based on FEA. Recently, some methods combining IGA have also appeared. [Li et al. (2021)] presents a *graph neural network* (GNN)-based deep learning model to learn the IGA-based material transport simulation and provide fast material concentration prediction within neurite networks of any topology. [Balu et al. (2019)] proposes a deep learning framework for the design and analysis of surgical bioprosthetic heart valves. This framework predicts the final deformations of the aortic valve and the valve coaptation area using the original undeformed geometry of the valve, the aortic pressure, and the material properties of the valve as input. This work performs the convolution operations on the control mesh of the input geometry, without any loss of geometric information, and obtains the displacement of each control point to obtain the deformation of the entire geometry. In this paper, we propose a more general framework for isogeometric analysis-reuse. A deep learning method combined with IGA is studied to solve PDEs on the topology-consistent B-spline parameterization with complex geometry.

3. Proposed method

In this section, we will present the framework of IGA-Reuse-Net, including the preliminaries, dataset generation and preprocessing, IGA-Reuse-Net architecture, construction of loss functions and setting of training parameters and performance indicators.

3.1. Preliminary

In this paper, we focus on solving the Poisson equation, which appears in a variety of problems, including heat conduction, gravity, simulation of fluid flow, and electrodynamics. The general form of the Poisson equation is defined as follows:

$$\begin{aligned} -\Delta\phi &= f, x \in \Omega \\ B\phi &= g, x \in \Gamma \subset \partial\Omega \end{aligned}, \tag{1}$$

where ϕ is an unknown function, Δ is a differential operator that represents the divergence of the gradient, f is the source function, B is a boundary operator, and g is a function defined on the boundary of the computational domain $\Omega \subset R^d$. $\partial\Omega$ represents its boundary, and Γ denotes the portion where the boundary conditions are applied.

In the framework of IGA, the computational domain Ω can be parametrized into G as

$$G(u, v) = \sum_{i \in I} \sum_{j \in J} R_{ij}(u, v) P_{ij} \tag{2}$$

where R_{ij} is tensor product of the bivariate basis function of parametric domain $\Omega_0 = [0, 1]^2$ [Cox (1972)], and $\{P_{ij}\}$ are the control points.

When using the IGA method for simulation analysis, we use the same set of basis functions $\{R_{ij}\}$ as those used in the modeling. The numerical solution of PDE over the computational domain Ω is then defined as

(a) Human Body Models with 15 Patches

(b) Flower Models with 5 Patches

(c) Genus-1 Models with 4 Patches

Fig. 1. Three sets of topology-consistent B-spline models. Different patches are rendered in different colors.

$$\phi_h(x, y) = \sum_{i \in I} \sum_{j \in J} R_{ij} \left(G^{-1}(x, y)\right) u_{ij} \tag{3}$$

where $\phi_h(x, y)$ is the numerical solution of the PDE, and $\{u_{ij}\}$ are the coefficients to be solved.

The purpose of our work presented in this paper is to quickly predict the numerical solution of the Poisson equation on a complex geometry with consistent topology via deep learning framework. Firstly, we solve the Poisson equation on a set of topology-consistent geometries to obtain the corresponding IGA numerical solutions. Then, we use the dataset from the geometries and numerical solutions to train a neural network. The learned model can directly predict the numerical solution of the Poisson equation on other topology-consistent geometries.

Topology-consistent parameterization means that the topologies of the geometries are the same but allow variants in different shapes. In our framework, the input of the neural network is the coordinates of the control points $\{P_{ij}\}$, and the output of the neural network is the coefficients $\{u_{ij}\}$ of the numerical solutions. If we denote the operation of the neural network as Z, it can be represented as a function as

$$\hat{U} = Z(P; \Theta) \tag{4}$$

where $P \in R^{M \times N}$, which includes the control points, and $\hat{U} \in R^{M \times N}$, which includes the coefficients predicted by the neural network. Then, the predicted numerical solution of the network can then be obtained as

$$\hat{\phi}_h(x, y) = \sum_{i \in I} \sum_{j \in J} R_{ij} \left(G^{-1}(x, y)\right) \hat{u}_{ij} \tag{5}$$

with $\{\hat{u}_{ij}\}$ extracted from $\hat{U}$.

3.2. Dataset generation and pre-processing

The method proposed in this paper is to train a CNN model in a data-driven manner. In order to generate the dataset for training, we need to make a set of models with consistent topology but large shape variations, and solve the Poisson equation on these models to obtain numerical solutions as labeled samples of a dataset. The detailed process is presented as follows.

Making a set of topology-consistent B-spline models In practice, we have generated three sets of 2D B-spline models with different topologies as shown in Fig. 1. The first set of models consists of 1575 human body models where each is composed of 15 bicubic B-spline patches with 4×4 control points as shown in Fig. 1(a). Each model is then randomly rotated into three other orientations for data augmentation. As a result, the final set contains 6300 models in total. The second set consists of 4000 flower models where each is composed of 5 bicubic B-spline patches with 5×5 control points as shown in Fig. 1(b). The third set of models consists of 5880 models with a hole (genus number as one) – see the examples shown in Fig. 1(c). Each model is composed of 4 bicubic B-spline patches with 5×5 control points.

Data normalization of B-spline models First, we need to normalize the B-spline models and limit them to the range of $[0, 1]^2$. We firstly find the minimum and maximum values of each model's x- and y-coordinates respectively, and calculate the scaling ratio $s = 1/\max\{x_{\max} - x_{\min}, y_{\max} - y_{\min}\}$ to obtain the normalized coordinates $(s(x - x_{\min}), s(y - y_{\min}))$.

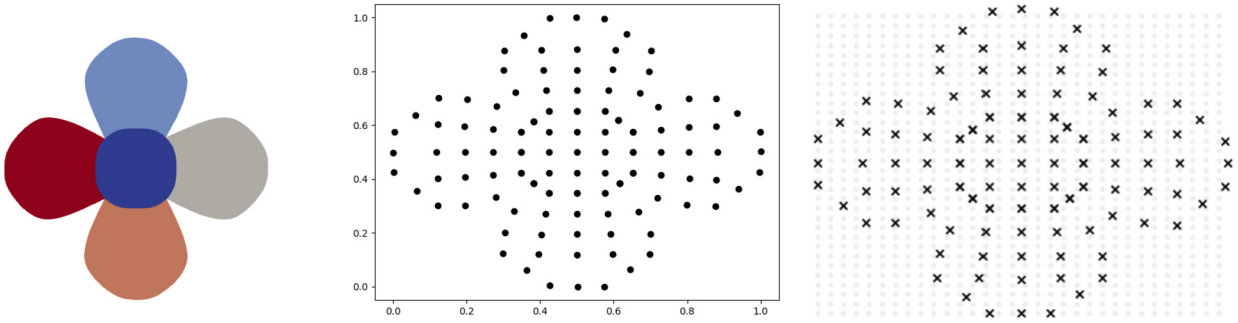


Fig. 2. Making the control points of the normalized B-spline model into a matrix.

Solving Poisson equation by IGA method After obtaining the normalized geometries, we solve the Poisson equation on these models, and obtain the IGA numerical solutions for each model. We perform h-refinement once on the parameterization of human body models by knot insertion. For the set of flower models, we perform h-refinement twice to obtain a representation with 605 degrees of freedom. For the models with 1-genus, h-refinement is performed twice to obtain a representation with 484 degree of freedoms.

After obtaining the normalized geometries and the IGA solutions, we convert them into the input and output format required by the neural network. The network architecture employed in this paper is a CNN model, hence the control points $\{P_{ij}\}$ of the B-spline models and the coefficients $\{u_{ij}\}$ in the IGA numerical solutions are converted into a matrix form as an image.

First, we extract the control points from the normalized B-spline models, as shown in the middle of Fig. 2 where the black dots represent the control points. Then according to a simple mapping $r_{ij} = \lfloor x_{ij} \times M \rfloor_{ij}, c_{ij} = \lfloor y_{ij} \times N \rfloor$, the subscripts (r_{ij}, c_{ij}) in the $M \times N$ matrix are obtained by the coordinates (x_{ij}, y_{ij}) of the control points P_{ij}. As shown in the right of Fig. 2, the black '**x**' indicates that these positions are control points, and the remaining light-gray dots indicate the zero-entries.

After obtaining the subscripts of all control points in the matrix, we fill in different values in the positions of the control points in the matrix to obtain matrices with different meanings. We fill 1 to obtain a $0-1$ matrix representing the positions of the control points, and fill in the x-coordinate of the control points to obtain the x-coordinate matrix; the y-coordinate matrix can be obtained in a similar way. There is a one-to-one correspondence between the control points and the coefficients, so we can fill in the coefficients' values to get the matrix of coefficients. The input data made by our method not only retains the basic shape of the geometry, but also the input geometry is accurate without any loss of information, because the input is the coordinates of the control points.

3.3. Network architecture

After the pre-processing steps presented in Section 3.2, the convolution operation can be used on the matrices of control points. Through multiple convolutions and nonlinear operations, a function Z can always be found, which represents the relationship between the control points P and the coefficients $\hat{U}$ of numerical solution, as shown in formula (4).

The CNN model used in this paper is based on the UNet3+ architecture [Huang et al. (2020)], which is a more powerful network architecture improved from UNet [Ronneberger et al. (2015)] and UNet++ [Zhou et al. (2018)]. UNet3+ is a deep learning network with encoder-decoder architecture, and uses full-scale skip connections to combine low-level details with high-level semantics from feature maps in different scales, making full use of the feature information extracted at each stage. UNet3+ not only improves the accuracy of network prediction, but also improves the computational efficiency.

As shown in Fig. 3, including the input layer and the output layer, the network used in this paper consists of 12 layers and corresponding convolution blocks. Each convolution block has a similar structure: convolutional calculation, batch-normalization, and a non-linear activate function. Among them, the Down-sampling operation includes max-pooling and a convolution block, and the Up-sampling operation includes the bilinear mode up-sample and a convolution block. The skip connections are represented by dotted lines in Fig. 3, and this module is composed of max-pooling or up-sample and a convolution block.

In addition to the classic UNet3+ architecture, we add a self-attention (SA) [Vaswani et al. (2017)] layer before the output layer. The SA mechanism, which is proposed to solve the problem of machine translation, has also been widely used in tasks based on the CNN architecture in recent years. Convolution is a local operation, so the CNN architecture needs to introduce SA to obtain the long-distance dependence of pixels to obtain more global features. In this paper, we introduce the ISSA module [Huang et al. (2019)], the accelerated SA module, which can effectively obtain global features and improve the accuracy of the network.

As shown in Fig. 3, the input size of the network is $128^2 \times 3$, and the output size is $128^2 \times 1$. We input the $0-1$ matrix, x-coordinate matrix, and y-coordinate matrix of the control points into the network, and then through the encoding

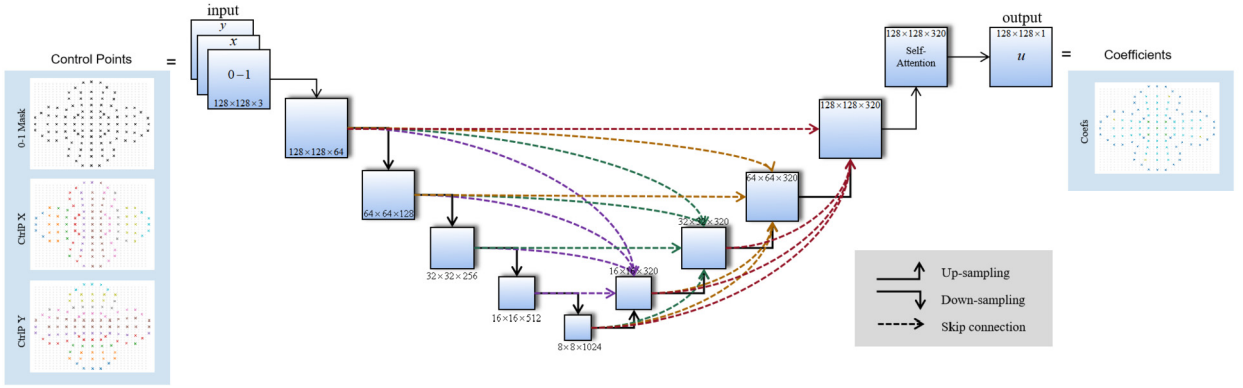


Fig. 3. Network Architecture.

and decoding process and skip connections operation, finally we can output the coefficients matrix. The coefficient values are extracted from the coefficients matrix to construct a smooth and continuous predictive numerical solution $\hat{\phi}_h$ for the Poisson equation as the formula (5).

3.4. Loss functions

MSE Loss In deep learning, the most commonly used loss function for regression tasks is the *mean square error* (MSE), which is the mean value of squared errors between the predicted coefficients and the corresponding ground-truth coefficients across all data points. Its formula is defined below:

$$L_{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{w}_i - w_i\right)^2 \tag{6}$$

where $\hat{w}_i$ is the output of the network, w_i is the ground truth, and n represents the number of data. In this paper, the value of n is 128^2.

Coefficients Loss In our work, the output coefficients matrix has a large number of zero-entries. However, all we need is the coefficients. Since the MSE loss includes all the zero-entries, it will make the network take a lot of effort to fit these zero-entries during training. In other words, directly using L_{MSE} as our optimization objective may harm the learning process. Hence, we propose a coefficients loss to measure the L1 distance between predicted coefficients and the ground truths. The coefficients loss is formulated as,

$$L_u = \frac{1}{m}\sum_{i=1}^{m}\left|\hat{u}_i - u_i\right| \tag{7}$$

where $\hat{u}_i$ denotes the predicted coefficient, u_i denotes the coefficient in the IGA numerical solution obtained by an IGA solver, and m is the number of coefficients, i.e. the degree of freedom in the IGA framework. L_u only calculates the error between the coefficients and doesn't consider the zero-entries. In this way, the neural network will concentrate on fitting the values of these coefficients, which is promising to improve the prediction accuracy.

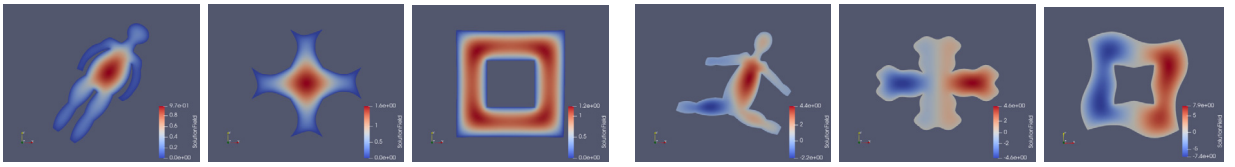
Numerical Solution Loss In addition, the predicted coefficients should be linearly combined with B-spline basis functions, and converted to a continuous and smooth numerical solution $\hat{\phi}_h$, as shown in formula (3). Therefore, in the training stage, we additionally use a numerical solution loss, i.e.

$$L_{\phi_h} = \frac{1}{S}\sum_{k=1}^{S}\left|\hat{\phi}_h\left(x_k, y_k\right) - \phi_h\left(x_k, y_k\right)\right| \tag{8}$$

where S represents the number of sampling points on the geometry. S is an important factor in the training of the network model. We therefore conduct a series of experiments to understand how S affects the performance. Corresponding results will be shown in Section 4.2.2.

Total Loss The aforementioned two loss functions L_u and L_{ϕ_h} focus on different aspects. Specially, L_u captures the characteristics of data and focuses on reducing the error of coefficients. In contrast, L_{ϕ_h} focuses on reducing the error of the numerical solution, which is specially designed for this task. Therefore, we combine these two loss functions for training. Our total loss is formulated as:

$$L_{total} = \alpha L_u + \beta L_{\Phi_h} \tag{9}$$



(a) IGA solution of Poisson equation with source function f_1 (b) IGA solution of Poisson equation with source function f_2

Fig. 4. Visual display of the numerical solutions of the Poisson equations with different source functions on the human body models, flower models and models with 1-genus. (a) corresponds to f_1, and (b) corresponds to f_2. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

where α and β are weighting factors, which are both set to be 1 as default in all experiments.

3.5. Training settings and performance indicators

In our experiments, we choose ReLU as the activation function, the Adam optimizer is used to train the network model, and the learning rate is set to 0.0006. The number of training iterations is 40k. The batch size is set to 8. The number of sampling points S of each patch is 400. The self-made dataset is divided at a ratio of 8:1:1 to obtain a training set, a validation set, and a test set. In the process of making B-spline models in Section 3.2, we first obtain some basic models, and then transform them to generate more models in batches. Therefore, when dividing the dataset, we divide the basic models and their transformed models into the same dataset, so the geometries in the verification set and the test set are not seen by the network model, nor are they transformed from the training set. This division method ensures the authenticity of the prediction of the network model.

For the learned models, we need to test the absolute error of the coefficients, the absolute error of the numerical solution, the relative error of the numerical solution, and the posterior error of the Poisson equation. The calculation of the absolute error of the coefficients and the absolute error of numerical solution is the same as formula (7) and formula (8) in Section 3.4, they can reflect the role of the loss function. The relative error of the numerical solution is the relative error between the solution of the network prediction and the IGA numerical solution. It is convenient to compare the effects of different network models. The calculation formula is defined as follows:

$$e_r = \frac{1}{S}\sum_{k=1}^{S}\left|\frac{\hat{\phi}_h(x_k, y_k) - \phi_h(x_k, y_k)}{\phi_h(x_k, y_k)}\right| \tag{10}$$

Furthermore, we can derive an explicit expression of the posterior error of the Poisson equation,

$$e_p = |\Delta\hat{\phi}(x, y) + f(x, y)| = \left|\frac{\partial^2\hat{\phi}(x, y)}{\partial x^2} + \frac{\partial^2\hat{\phi}(x, y)}{\partial y^2} + f(x, y)\right| \tag{11}$$

where $f(x, y)$ is the source function in Poisson equation. The posterior error of PDEs can indicate whether the predicted solution can satisfy the laws of PDEs.

4. Experiments and discussions

In this section, we will show some experimental results and give some comparisons and discussions.

We solve two kinds of Poisson equations with different source functions $f(x, y)$ on the three sets of different geometries:

$$\begin{aligned} &-\Delta\phi(x, y) = f(x, y), (x, y) \in \Omega \\ &\phi(x, y) = 0, (x, y) \in \partial\Omega \end{aligned}, \tag{12}$$

where $\Omega \subset R^2$, and the numerical solutions on the boundaries of the geometries are all designated as 0.

The first source function is a constant function $f_1 = 100$, and the IGA numerical solution of this Poisson equation is shown in Fig. 4(a). We call the datasets made by solving the Poisson equation with the source function f_1 on the human body models, flower models, models with 1-genus as $human\_f_1$, $flower\_f_1$ and $hole\_f_1$ respectively.

The second source function is a periodically changing function $f_2 = -100 \times \pi^2 \times \sin(2 \times \pi \times x)$, and the corresponding IGA numerical solution of Poisson equation is shown in Fig. 4(b). The datasets made by solving the Poisson equation with the source function f_2 on the human body models, flower models, models with 1-genus are denoted as $human\_f_2$, $flower\_f_2$ and $hole\_f_2$ respectively.

Table 1
Comparison of prediction errors of UNet and UNet3+ trained on $human\_f_1$ and $flower\_f_1$.

Dataset	Network Architecture	Absolute Error of Coefficients	Absolute Error of Solution	Relative Error of Solution (%)	Posterior Error
$human\_f_1$	UNet	0.0385	0.0170	12.02	54.84
	UNet3+	0.0206	0.00952	7.05	37.34
$flower\_f_1$	UNet	0.0273	0.0163	4.06	37.70
	UNet3+	0.0144	0.00892	2.38	24.23

Table 2
Verification of the effect of the ISSA module on $human\_f_2$ and $flower\_f_2$.

Dataset	model	Absolute Error of Coefficients	Absolute Error of Solution	Relative Error of Solution (%)	Posterior Error
$human\_f_2$	UNet3+	0.130	0.0639	17.80	219.2
	ISSA + UNet3+	0.125	0.0583	16.06	213.9
$flower\_f_2$	UNet3+	0.0876	0.0572	12.10	140.8
	ISSA + UNet3+	0.0860	0.0553	10.95	143.4

4.1. Experiments and discussions about network architecture

4.1.1. UNet vs. Unet3+

First, we compare UNet3+ with UNet, which is a strong baseline in various computer vision tasks. The UNet architecture follows the encoder-decoder architectures and has shown surprising performance in a large number of image-to-image translation tasks. UNet3+ improves UNet by adding full-scale skip connections for exploring more information. To verify our motivation of using UNet3+ in our task, we train the UNet model and UNet3+ model on datasets $human\_f_1$ and $flower\_f_1$, respectively. Note that, including the input and output layers, UNet is composed of 15 layers, UNet3+ has 12 layers; and all the other experimental settings are the same.

We use the aforementioned four indicators (in Section 3.5) to evaluate performances of learned UNet and UNet3+ models on the testing set. The corresponding results are reported in Table 1. Obviously, UNet3+ model performs much better than UNet. On both test sets, UNet3+ achieves distinctly lower values of all the four test indicators, in contrast to UNet. Specially, UNet3+ reduces the value of each indicator by about 50%. Such achievements are mainly caused by the full-scale skip connections in UNet3+, which are significant for dense predictions in our task. Notably, UNet also shows inspiring performance in this task. The relative errors of the numerical solutions predicted by UNet are 4.06% and 12.02% on the $human\_f_1$ and $flower\_f_1$ test sets, respectively. Besides, UNet achieves low values in terms of the other three indicators.

In Fig. 5, we visualize the predicted solutions of UNet and UNet3+ on the two test sets, as well as their absolute and relative errors with IGA numerical solution. The visualization results of the first three columns in Fig. 5 are very close, indicating that both UNet and UNet3+ are suitable for our task. However, from the pairwise comparisons in the last four columns, there are more red areas (i.e. large errors) in the error map related to UNet, in contrast to the error map related to UNet3+. This comparison result indicates that the predictions of UNet3+ are more accurate than UNet.

Based on both the quantitative comparison (i.e. numerical performance indicators) and the qualitative results (i.e. the visualization images), we find that UNet3+ is superior in our task than UNet, demonstrating our motivation of using UNet3+ as the baseline.

4.1.2. Effect of ISSA

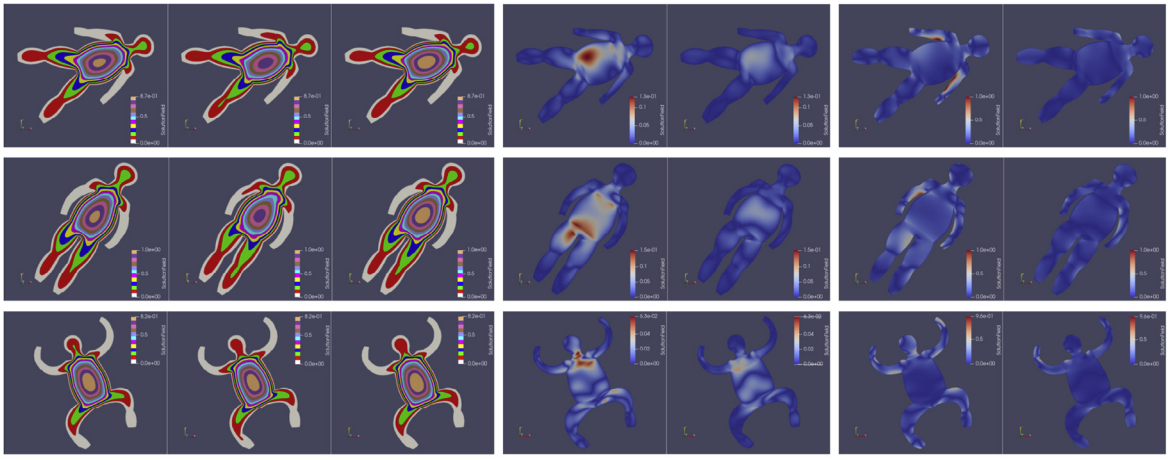
Although UNet3+ itself has shown excellent performance in our task, as reported previously, it lacks the long-distance dependence between different positions. Therefore, we use the ISSA module to globally model correlations between different correlations.

To analyze the impact of ISSA, based on the UNet3+ architecture, we perform experiments with and without ISSA module on $human\_f_2$ and $flower\_f_2$, respectively. The corresponding results are shown in Table 2. As expected, the ISSA module decreases all the four indicators on both test sets. Specially, relative errors of the numerical solution decrease from 17.80% and 12.10% to 16.06% and 10.95%, respectively, and the decrease ratio is about 9.5%. From the perspective of deep learning, such improvement is typically considered noticeable. These comparison results demonstrate that the ISSA module can effectively enhance the predictive ability of UNet3+.

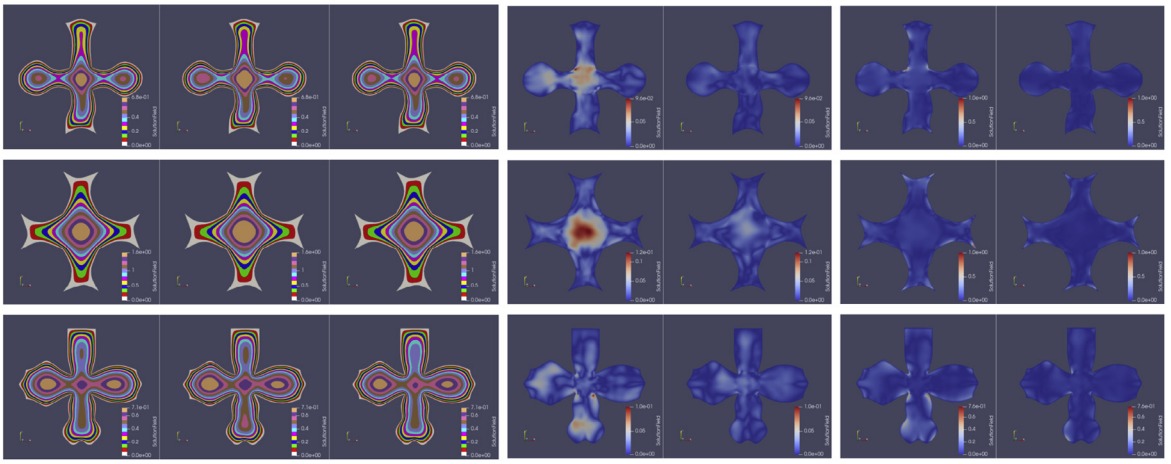
4.2. Experiments and discussions about loss functions

4.2.1. Loss verification

In this paper, our full loss function, L_{total}, is a combination of the coefficients loss, L_u, and numerical solution loss, L_{ϕ_h}. In this section, we train our full model by separately using different loss functions in Section 3.4, and evaluate the learned models correspondingly, to verify the effectiveness of L_{total}. Specially, we use four loss functions, i.e. L_{MSE}, L_u, L_{ϕ_h}, and L_{total}, to train our full model on $human\_f_1$ and $flower\_f_1$, respectively. The errors of corresponding learned models on test sets are shown in Table 3.



(a) Analysis-reuse on $human\_f_1$



(b) Analysis-reuse on $flower\_f_1$

Fig. 5. Visualization of the predicted analysis-reuse results of UNet models and UNet3+ models trained on $human\_f_1$ and $flower\_f_1$. From left to right, each row is: IGA numerical solution ϕ_h, UNet predicted solution ϕ_U^h, UNet3+ predicted solution ϕ_{U3}^h, absolute error between ϕ_U^h and ϕ_h, absolute error between ϕ_{U3}^h and ϕ_h, relative error between ϕ_U^h and ϕ_h, relative error between ϕ_{U3}^h and ϕ_h.

Table 3
Comparison of the four loss functions on $human\_f_1$ and $flower\_f_1$.

Dataset	Loss Function	Absolute Error of Coefficients	Absolute Error of Solution	Relative Error of Solution (%)	Posterior Error
$human\_f_1$	L_{MSE}	0.0324	0.0151	13.00	53.93
	L_u	0.0213	0.00983	7.67	36.56
	L_{ϕ_h}	0.0289	0.0103	7.31	38.62
	L_{total}	0.0206	0.00952	7.05	37.34
$flower\_f_1$	L_{MSE}	0.0272	0.0170	4.78	39.07
	L_u	0.0167	0.0114	2.69	23.96
	L_{ϕ_h}	0.0170	0.00898	2.40	29.00
	L_{total}	0.0144	0.00892	2.38	24.23

First, when using L_{MSE} to train the network, the values of all performance indicators are low, as shown in Table 3. Specially, the relative errors of numerical solutions on these two test sets are 13% and 4.70%. In contrast to L_{MSE}, using L_u to train the network will further reduce all errors by about $30\% \sim 40\%$, relatively. The reason why L_u achieves such performance improvements is that L_u only focuses on the error of coefficients in training. However, the error of zero-entries is calculated in L_{MSE}, which may disturb the learning process.

When training the network with L_{ϕ_h}, similar to L_u, all four errors are reduced by about $30\% \sim 40\%$. Such performance improvements indicate that the numerical solution loss, which is specially designed for the task, is rather effective for guiding the training of the network.

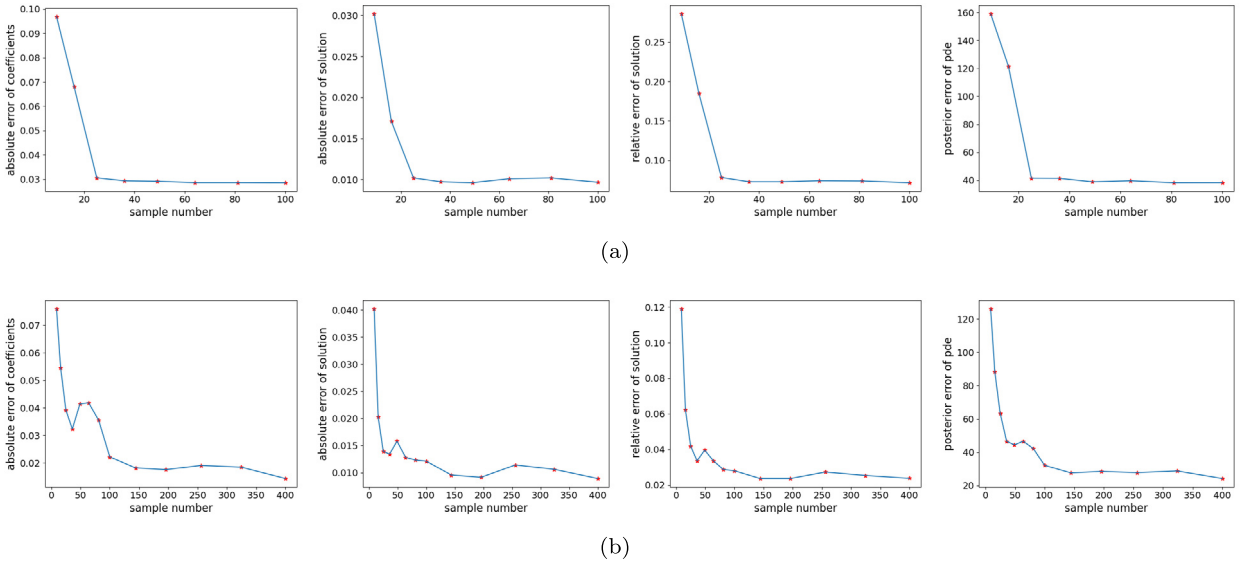


Fig. 6. Influences of the number of sampling points on performance, when the model is trained with L_{ϕ_h}. (a) Curves of four performance indicators vs. S on $human\_f_1$; and (b) Curves of four performance indicators vs. s on $flower\_f_1$. The abscissa of each picture is S, and the ordinate is the test indicator. The ordinates of each row from left to right are: absolute error of coefficients, absolute error of numerical solution, relative error of numerical solution, and posterior error of Poisson equation.

Since L_u and L_{ϕ_h} captures the characteristics of data and those of the task respectively, they have different impacts on performance. As shown in Table 3, the absolute error of coefficients related to L_u is lower than that related to L_{ϕ_h}; while L_{ϕ_h} leads to lower absolute and relative errors of the numerical solution, in contrast to L_u. As expected, if we train the network by using L_{total}, i.e. combining L_u and L_{ϕ_h}, we will obtain the lowest errors in terms of all the four test indicators. This achievement indicates that combining L_u and L_{ϕ_h} not only doesn't affect each other's training process, but also integrates their advantages. In other words, the benefits of L_u and L_{ϕ_h} are at least partially additive.

4.2.2. Influence of the number of sampling points

When introducing the loss function L_{ϕ_h} in Section 3.4, we mentioned that the number of sampling points S has an impact on training the network with L_{ϕ_h}. In this section, we conduct experiments to study the influence of the number of sampling points, S, on the performance of the network model with L_{ϕ_h} as the learning objective.

We conduct experiments on $human\_f_1$ and $flower\_f_1$. For human body models, we sample points uniformly on each spline. There are 9 choices of sampling numbers, from 2^2 to 10^2. For flower models, there are 19 choices of numbers, from 2^2 to 20^2. Correspondingly, we set different S, and perform 9 experiments on $human\_f_1$ and 19 experiments on $flower\_f_1$. The corresponding results are shown in Fig. 6.

From Fig. 6, we can observe that: for different datasets and different test indicators, the broken lines first decline and then become stable. This trend indicates that if S increases within a certain range, the prediction errors decrease. However, beyond this range, increasing S will not affect the network prediction ability. The threshold is about 25 in the human body dataset, and about 144 in the flower dataset, which are basically the same as the number of coefficients to be fitted for each spline. In other words, when we train the network with L_{ϕ_h}, S of each spline must be no less than the number of coefficients to be fitted, so as to ensure the best training results.

4.3. Analysis-reuse results with IGA-Reuse-Net

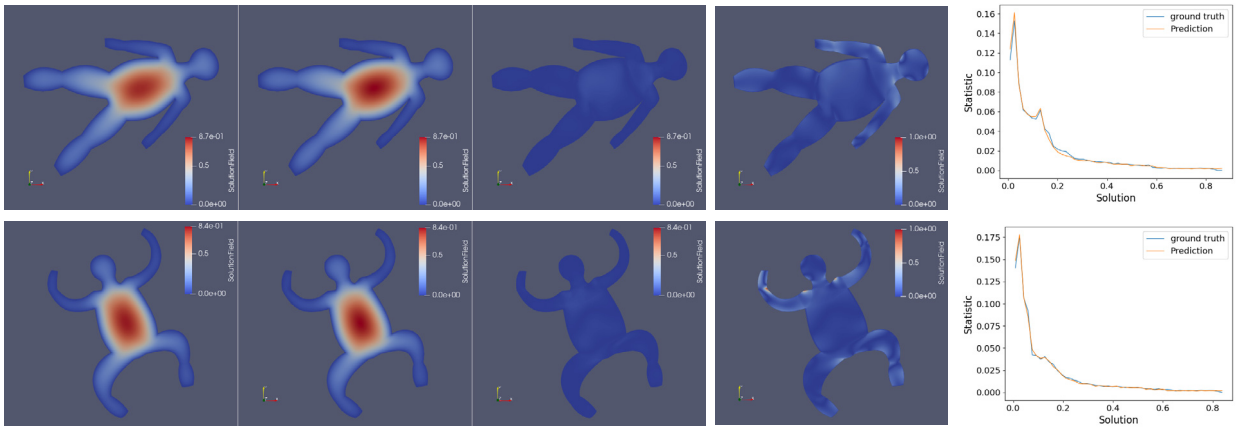
Two kinds of Poisson equations with different source functions have been solved on the human body models and the flower models, and produced four datasets $human\_f_1$, $flower\_f_1$, $human\_f_2$, and $flower\_f_2$. These two functions are $f_1 = 100$ and $f_2 = -100 \times \pi^2 \times \sin(2 \times \pi \times x)$. In this section, the IGA-Reuse-Net method is used to implement the analysis-reuse for these PDE problems, which are used to prove the effectiveness of the method in this paper.

4.3.1. Example 1

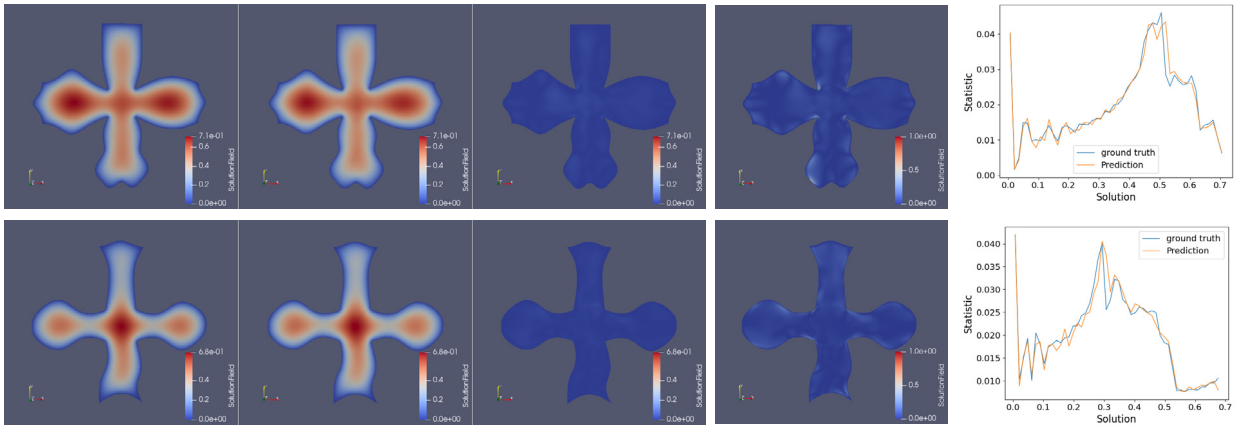
First, we use the method in this paper to solve the Poisson equation with $f_1 = 100$ as the source function. We train the network of this paper on $human\_f_1$ and $flower\_f_1$ respectively, and then use the optimal network models to predict the solution of the Poisson equation on other geometries that have not been seen by the model before. The values of the test indicators of the predictions are shown in the first part of Table 4. The visualization of the predicted solution and its errors with IGA solution is shown in Fig. 7.

Table 4
Performance of IGA-Reuse-Net for the human body dataset and flower dataset.

Source function	Dataset	Absolute Error of Coefficients	Absolute Error of Solution	Relative Error of Solution (%)	Posterior Error
$f_1 = 100$	$human\_f_1$	0.0206	0.00952	7.05	37.34
	$flower\_f_1$	0.0144	0.00892	2.38	24.23
$f_2 = -100 \times \pi^2 \times \sin(2 \times \pi \times x)$	$human\_f_2$	0.125	0.0583	16.06	213.9
	$flower\_f_2$	0.086	0.0553	10.95	143.4



(a) $human\_f_1$

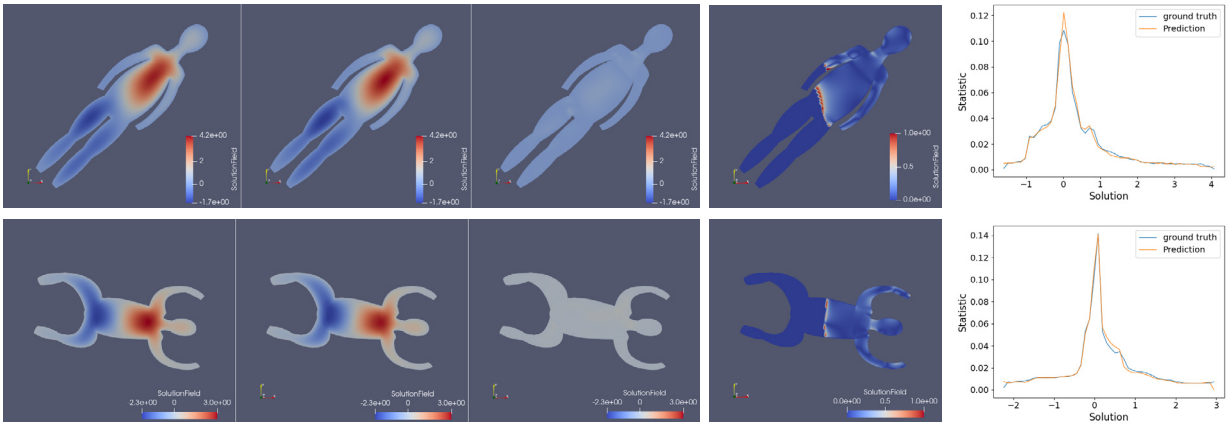


(b) $flower\_f_1$

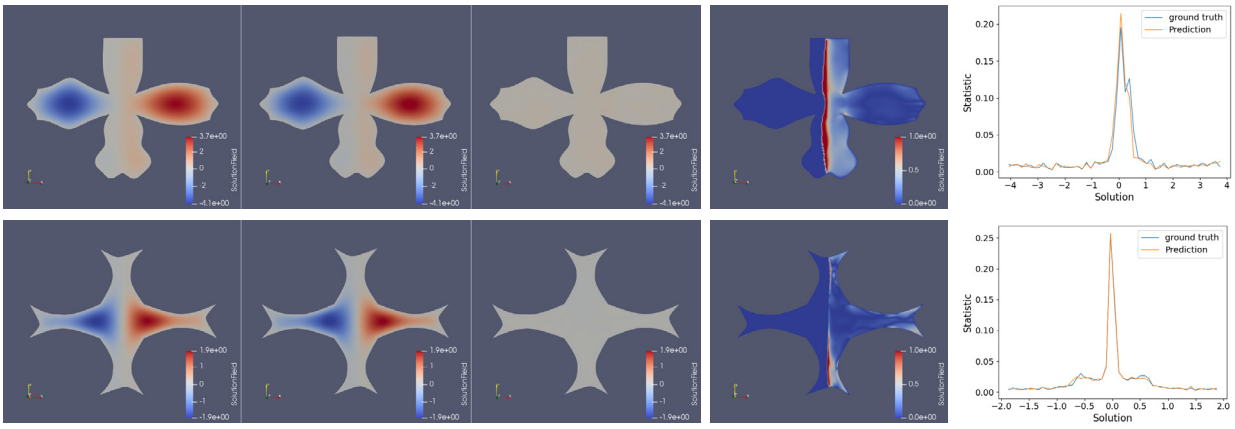
Fig. 7. Visualization of analysis-reuse on example 1. Each row from left to right is: IGA numerical solution ϕ_h, network predicted solution $\hat{\phi}_h$, absolute error and relative error between ϕ_h and $\hat{\phi}_h$, and statistical curves of ϕ_h and $\hat{\phi}_h$.

From Table 4, we can find that the network model trained on $human\_f_1$ has a relative error of 7.05% of the predicted solution; the network model trained on $flower\_f_1$ has a relative error of only 2.38%, which shows that our method can accurately predict the numerical solution of the equation. However, we can find that the prediction errors of the human body model are larger than that of the flower model, which is because as the number of patches increases, the model becomes more complex and the learning of the network is more difficult.

We also show several visualization results of predicted solutions and their errors, as shown in Fig. 7. The shape of the network predicted solution $\hat{\phi}_h$ in the second column and the IGA solution ϕ_h in the first column are very similar, and the error value of $\hat{\phi}_h$ is very small, indicating that the network in this paper can accurately approximate ϕ_h. In addition, we sample and count the numerical solutions ϕ_h and $\hat{\phi}_h$ respectively to provide the statistical results, as shown in the last column of Fig. 7. The blue curve is the statistical result of ϕ_h, and the orange curve is the statistical result of $\hat{\phi}_h$. The two statistical curves are also very close. From this point of view, it shows that the numerical solutions of the two are very similar. The fourth column is the relative error between ϕ_h and $\hat{\phi}_h$.



(a) $human\_f_1$



(b) $flower\_f_1$

Fig. 8. Visualization of analysis-reuse on example 2. Each row from left to right is the same as Fig. 7.

4.3.2. Example 2

Similarly, we also use the proposed method to solve the Poisson equation with the periodically changing function $f_2 = -100 \times \pi^2 \times \sin(2 \times \pi \times x)$ as the source function. We train the network of this paper on $human\_f_2$ and $flower\_f_2$ respectively, and then use the optimal network model to predict the solution of the Poisson equation on other geometries that have not been seen by the model before. The values of the test indicators of the predicted solution are shown in the second part of Table 4. The visualization of the predicted solution and its error with the IGA numerical solution is shown in Fig. 8.

From Table 4, the network model trained on $human\_f_2$ has a relative error of 16.06%; the network model trained on $flower\_f_2$ has a relative error of 10.95%. We can find that these errors are much higher than the experimental results on f_1, which is because the source function f_2 in this example is more complicated than f_1. We additionally calculate the posterior error of the ground-truth IGA numerical solutions. The posterior errors are 17.7, 4.85, 103.4, and 26.26, corresponding to the datasets shown in Table 4, from top to bottom. Obviously, even the performance of the ground-truth IGA numerical solution algorithm diverses greatly among different equations. And there are more numerical solutions close to 0, so the relative error values are all higher than example 1. We can also draw this conclusion from the visualization results Fig. 8.

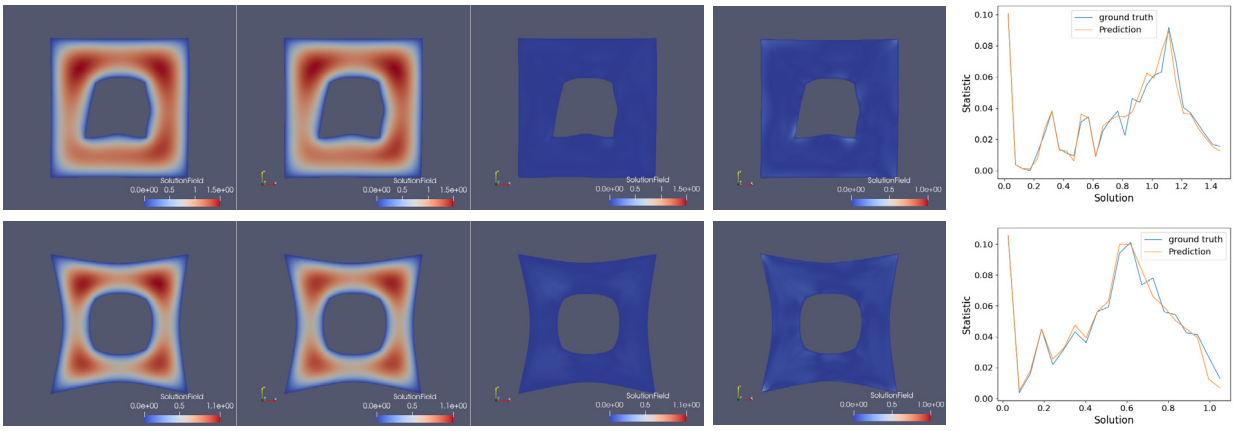
The visualization results of the predicted solution and its error are shown in Fig. 8. The shape of the network predicted solution $\hat{\phi}_h$ in the second column and the IGA solution ϕ_h in the first column are very similar, and the statistical curves of $\hat{\phi}_h$ and ϕ_h in the last column basically coincide, indicating that this method can predict the numerical solution of the equation in this example well.

4.4. Analysis-reusing on geometries with 1-genus

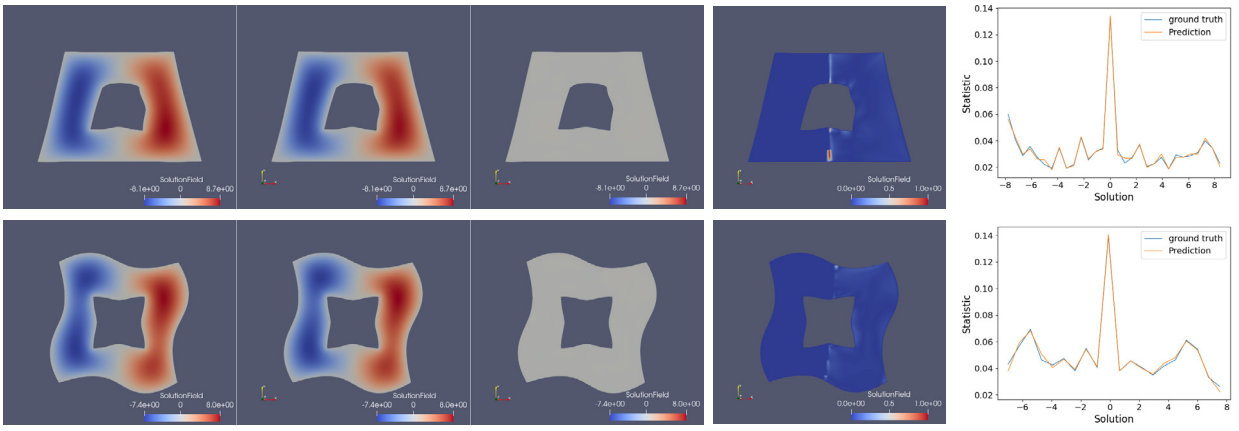
In order to verify the generality of our method, in this section, the proposed method is used to realize analysis-reuse on geometries with 1-genus. Firstly, we make a set of geometries with 1-genus, and as shown in Fig. 1(c), each geometry has a

Table 5
Performance of IGA-Reuse-Net on the geometries with 1-genus.

Dataset	Absolute Error of Coefficients	Absolute Error of Solution	Relative Error of Solution (%)	Posterior Error
*hole_*f_1	0.0148	0.0108	1.42	12.63
*hole_*f_2	0.0878	0.0594	3.63	91.7



(a) *hole_*f_1



(b) *hole_*f_1

Fig. 9. Visualization of the predicted analysis-reuse solution of Poisson equation with f_1 and f_2 as source functions on geometries with 1-genus. Each row from left to right is the same as Fig. 7.

hole inside. Similarly, two kinds of Poisson equations with different source functions on these geometries with 1-genus are solved, and two datasets *hole_*f_1 and *hole_*f_2 can be generated. The values of the test indicators of predicted solution are shown in Table 5. The visualization of the predicted solution and its errors with the IGA solution is shown in Fig. 9.

From Table 5, when solving the Poisson equation with f_1 as the source function, the relative error of the IGA-Reuse-Net predicted solution is 1.42%; when solving the Poisson equation with f_2 as the source function, the relative error of the IGA-Reuse-Net predicted solution is 3.63%. Especially when the f_2 as the source function, the relative error of the prediction solution on the geometries with 1-genus is much lower than that of the human body models and the flower models, which has a lot to do with the complexity of the geometries. In addition, as shown in Fig. 9(b), the visualization of the relative error in the fourth column, we can find that the relative error of the geometries with 1-genus is much lower than that on the human body models and the flower models in Fig. 8.

We realize the analysis-reuse for two kinds of Poisson equations with different source function on this set of geometries with 1-genus. Our method can quickly and accurately predict the solution of Poisson equation on geometries with different numbers of spline patches, and it can also predict on geometries with a hole inside. Besides, our current CNN architecture can also be extended to relatively regular 3D geometries, i.e. the control mesh of 3D geometry that can be easily sliced. But we cannot generate the matrix format dataset of complex 3D geometries by using our mapping method. In the future, we will make efforts to extend the proposed methodology to volumetric parameterization.

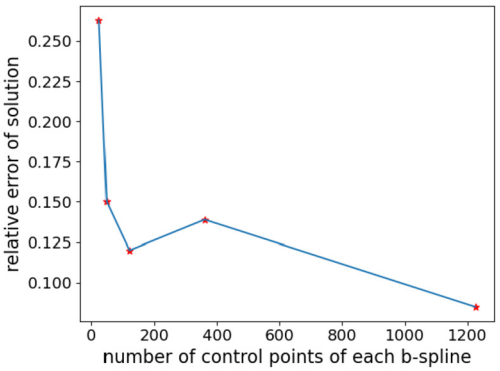


Fig. 10. Our network is trained on datasets with different numbers of control points, and the relative errors of each network model on the test data.

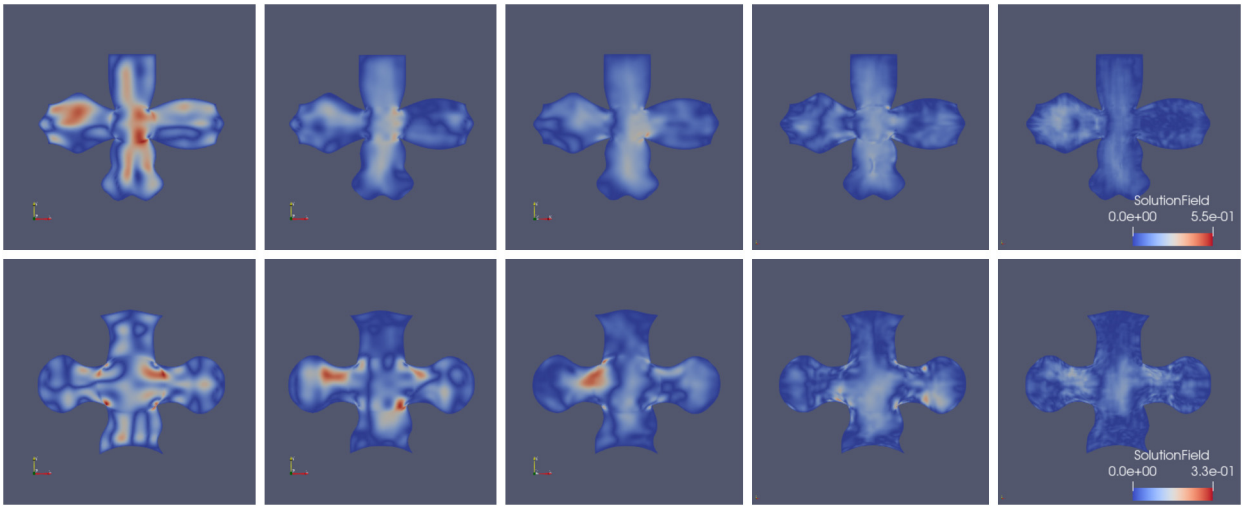


Fig. 11. Our network is trained on datasets with different numbers of control points. The visualization of the errors between the prediction results of the trained network model and the IGA numerical solution refined 4 times. From left to right are the errors of refinement 0 to 4 times.

4.5. The influence of the number of control points on network training

In order to change the number of control points of the b-spline model, we uniformly refine the flower model by inserting knots, and a total of 5 sets of data with different numbers of refinement are generated. We refine the flower model 0 to 4 times, and the number of control points on each b-spline is 5^2, 7^2, 11^2, 19^2, and 35^2, and the number of patches are 4, 16, 64, 256 and 1024. When transforming the data refined 4 times into matrix format, since the matrix of 128^2 is not enough to hold so many control points, we transform the data into matrix format of 256^2. We train our network on these datasets with different times of refinement, and compare the relative errors of the numerical solutions of each network model on the test data, as shown in Fig. 10. Note that the numerical solutions compared during the test of each network model are numerical solutions at their respective precisions.

From Fig. 10, we can draw some conclusions. From refinement 0 to refinement twice, as the number of control points increases, the relative error decreases significantly. This phenomenon is related to the characteristics of the dataset in our paper. We make the input matrix with control points and padding 0s, and only the control points are useful values. For the original model without refinement, there are only 125 control points of the entire geometry. Compared with 128^2 pixels, the proportion of these control points is very small, so other padding 0s have a great impact on network learning. However, as the number of control points increases, the useful data increases. This is beneficial for network learning, so the relative error decreases. The relative error of three times of refinement fluctuates, which is because the data fed to the network is still $128^2 \times 3$, but the number of coefficients to be fitted by the network has increased a lot. But the relative error of four times of refinement has dropped a lot, because the size of the matrix we feed to the network is $256^2 \times 3$. Compared with the matrix of $128^2 \times 3$, the amount of data refined four times is increased four times, and the useful value is also more, so it is very beneficial to the network learning.

In addition, as the number of refinement increases, the precision of the IGA numerical solution also increases. As shown in Fig. 11, we compare the prediction results of the previously trained network model with the IGA numerical solution

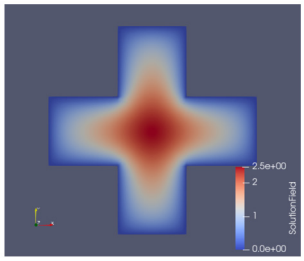


Fig. 12. Regular geometry with five B-spline patches.

Table 6
Comparison of the performance between our method and PINNs in solving Poisson's equation.

Method	Absolute Error of Solution	Relative Error of Solution (%)	Time(s)
Our Method	0.0583	11.97	0.013
PINNs 5K	0.137	20.5	867
PINNs 10K	0.0765	14.3	1714
PINNs 15K	0.0550	11.08	2631

refined 4 times. It can be found that as the accuracy of the dataset increases, the accuracy of the network prediction also increases.

4.6. Comparison with PINNs

We compare the performance of the proposed method with the PINNs [Raissi et al. (2019)] that have been studied a lot in recent years. The network architecture of PINNs is very simple, it is a fully connected network. The input of PINNs is the coordinates (x_i, y_i) of the sampling point on the geometry, and the output is the numerical solution $\hat{\phi}(x_i, y_i)$ corresponding to this point. PINNs can only solve PDEs on certain geometry once, that is, for different geometry of computational domain, PINNs need to be trained before they can predict the solution of the entire computational domain. On the contrary, our method only needs to train the network once, then we can predict the solutions for different geometry with consistent topology. It takes about 1 hour to prepare a training data, and 5 hours to train our model. After the network training is completed, our method only takes 0.013 s to predict the solution of a topology-consistent geometry. Moreover, PINNs can only obtain discrete finite-element solution, and our method can obtain continuous and smooth spline numerical solution. Specially, PINNs take the coordinates of a number of sampling points on the geometry as input; and output the solution values of these sampling point. Thus by using PINNs, we can only obtain discrete numerical solutions. In contrast, our method take coordinates of all the control points of a geometry as input; and outputs the coefficients of these control points. Afterwards, we can construct an expression for the numerical solution over the entire geometry by linearly combining these coefficients with spline basis functions. Therefore we can obtain smooth and continuous solutions by using our proposed method.

For the comparison, we solve the Poisson equation on the flower model composed of five B-spline patches. The source function is $f_1 = 100$, and Dirichlet condition of 0 is imposed on the boundary. DEEPXDE [Lu et al. (2021)] library is used to train PINNs. The PINNs have 4 hidden layers with 50 neurons. We sample 2000 points on the geometry as shown in Fig. 12 for training, use Adam as the optimizer, and set the learning rate to be 0.001. The test results of the prediction and training time of PINNs with different iteration times are shown in Table 6.

From Table 6, we can find that as the number of training iterations increases, the test error of PINNs continues to decrease, but the training time continues to increase. Until the number of training iterations reaches 15k, the absolute error and relative error of the predicted solution of PINNs are lower than our method. At this time, the training time required for PINNs is 2631 s, and our method only needs 0.013 s to get a solution with similar accuracy. With the increase of test geometry, the test time required for PINNs increases linearly on the basis of 2631 s, while our test time only takes a few minutes at most. This experiment is sufficient to show that a good trade-off between accuracy and efficiency can be achieved by the proposed framework. Our method outperforms the PINN model when predicting the solution of the Poisson equation on a new geometry, because our method is not only fast, but also gets a continuous and smooth spline solution.

5. Conclusion

In this paper, we proposed a deep learning framework combined with isogeometric analysis called IGA-Reuse-Net for efficient reuse of numerical simulation on a set of topology-consistent B-spline planar parameterizations. Compared with previous data-driven numerical simulation methods only for simple computational domains, our method can predict more accurate PDE solutions over topology-consistent geometries with complex boundaries. A good trade-off between accuracy

and efficiency can be achieved by the proposed framework. Our framework outperforms the PINN model and yields remarkable prediction solutions.

Moreover, although the current research focus on solving the Poisson equation, the method proposed in this paper provides an idea to find a faster and more robust method to solve other complex PDEs.

CRediT authorship contribution statement

Dandan Wang: Conceptualization, Methodology, Software. **Jinlan Xu:** Methodology, Writing – original draft. **Fei Gao:** Investigation, Visualization. **Charlie C.L. Wang:** Conceptualization, Writing – review & editing. **Renshu Gu:** Software, Validation. **Fei Lin:** Methodology, Validation. **Timon Rabczuk:** Investigation, Writing – review & editing. **Gang Xu:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors would like to thank the anonymous reviewers for providing constructive suggestions and comments, that contributed to highly improve this work. This work was supported by the National Key R&D Program of China under Grant No. 2020YFB1709402, the NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Informatization (Grant No. U1909210), the National Natural Science Foundation of China under Grants 61971172 and 62072148, the Zhejiang Provincial Science and Technology Program in China under Grant No. 2021C01108 and No. 2020C01074, Zhejiang Lab Tianshu Open Source AI Platform, the Zhejiang Provincial Science and Technology Program in China under Grant No. LQ22F020026, and Fundamental Research Funds for the Provincial Universities of Zhejiang (Grant No. GK219909299001-028).

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al., 2016. Tensorflow: large-scale machine learning on heterogeneous distributed systems. arXiv preprint. arXiv:1603.04467.

Anitescu, C., Atroshchenko, E., Alajlan, N., Rabczuk, T., 2019. Artificial neural network methods for the solution of second order boundary value problems. Comput. Mater. Continua 59, 345–359.

Balu, A., Nallagonda, S., Xu, F., Krishnamurthy, A., Hsu, M.C., Sarkar, S., 2019. A deep learning framework for design and analysis of surgical bioprosthetic heart valves. Sci. Rep. 9, 1–12.

Beck, C., Weinan, E., Jentzen, A., 2019. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. J. Nonlinear Sci. 29, 1563–1619.

Cox, M.G., 1972. The numerical evaluation of b-splines. IMA J. Appl. Math. 10, 134–149.

Duraisamy, K., Iaccarino, G., Xiao, H., 2019. Turbulence modeling in the age of data. Annu. Rev. Fluid Mech. 51, 357–377.

Gao, H., Sun, L., Wang, J.X., 2020. Phygeonet: physics-informed geometry-adaptive convolutional neural networks for solving parametric pdes on irregular domain. arXiv e-prints. arXiv:2004.13145 [eess.IV].

Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. Neural Netw. 2, 359–366.

Huang, H., Lin, L., Tong, R., Hu, H., Wu, J., 2020. Unet 3+: a full-scale connected unet for medical image segmentation. arXiv:2004.08790 [eess.IV].

Huang, L., Yuan, Y., Guo, J., Zhang, C., Chen, X., Wang, J., 2019. Interlaced sparse self-attention for semantic segmentation. arXiv preprint. arXiv:1907.12273.

Hughes, T.J., Cottrell, J.A., Bazilevs, Y., 2005. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. Comput. Methods Appl. Mech. Eng. 194, 4135–4195.

Ketkar, N., 2017. Introduction to pytorch. In: Deep Learning with Python. Springer, pp. 195–208.

Lagaris, I.E., Likas, A., Fotiadis, D.I., 1998. Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans. Neural Netw. 9, 987–1000.

Li, A., Chen, R., Farimani, A.B., Zhang, Y.J., 2020. Reaction diffusion system prediction based on convolutional neural network. Sci. Rep. 10, 3894.

Li, A., Farimani, A.B., Zhang, Y.J., 2021. Deep learning of material transport in complex neurite networks. Sci. Rep. 11, 1–13.

Ling, J., Kurzawski, A., Templeton, J., 2016. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. J. Fluid Mech. 807, 155–166.

Lu, L., Meng, X., Mao, Z., Karniadakis, G.E., 2021. Deepxde: a deep learning library for solving differential equations. SIAM Rev. 63, 208–228.

Raissi, M., 2018. Forward-backward stochastic neural networks: deep learning of high-dimensional partial differential equations. arXiv preprint. arXiv:1804.07010.

Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019. 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.

Ramuhalli, P., Udpa, L., Udpa, S.S., 2005. Finite-element neural networks for solving differential equations. IEEE Trans. Neural Netw. 16, 1381–1392.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. Springer, Cham.

Thuerey, N., Weißenow, K., Prantl, L., Hu, X., 2020. Deep learning methods for Reynolds-averaged Navier–Stokes simulations of airfoil flows. AIAA J. 58, 25–36.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention is all you need. arXiv:1706.03762 [cs.CL].

Xu, G., Kwok, T.H., Wang, C.C., 2017. Isogeometric computation reuse method for complex objects with topology-consistent volumetric parameterization. Comput.-Aided Des. 91, 1–13.

Yao, H., Ren, Y., Liu, Y., 2019. Fea-net: a deep convolutional neural network with physicsprior for efficient data driven pde learning. In: AIAA Scitech 2019 Forum, p. 0680.

Zhou, Z., Siddiquee, M.M.R., Tajbakhsh, N., Liang, J., 2018. Unet++: a nested u-net architecture for medical image segmentation. In: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support. Springer, pp. 3–11.