Journal of Computational Physics 530 (2025) 113902

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A multiple transferable neural network method with domain decomposition for elliptic interface problems

Tianzheng Lu [a], Lili Ju [b], Liyong Zhu [a,*]

[a] School of Mathematical Sciences, Beihang University, Beijing 100191, China
[b] Department of Mathematics, University of South Carolina, Columbia, SC 29208, USA

ARTICLE INFO

Keywords:
Two-layer neural networks
Elliptic interface problems
Transferability
Domain decomposition
Neuron shape

ABSTRACT

The transferable neural network (TransNet) is a two-layer shallow neural network with predetermined and uniformly distributed neurons in the hidden layer, and the least-squares solvers can be particularly used to compute the parameters of its output layer when applied to the solution of partial differential equations. In this paper, we integrate the TransNet technique with the nonoverlapping domain decomposition and the interface conditions to develop a novel multiple transferable neural network (Multi-TransNet) method for solving elliptic interface problems, which typically contain discontinuities in both solutions and their derivatives across interfaces. We first propose an empirical formula for the TransNet to characterize the relationship between the radius of the domain-covering ball, the number of hidden-layer neurons, and the optimal neuron shape. In the Multi-TransNet method, we assign each subdomain one distinct TransNet with an adaptively determined number of hidden-layer neurons to maintain the globally uniform neuron distribution across the entire computational domain, and then unite all the subdomain TransNets together by incorporating the interface condition terms into the loss function. The empirical formula is also extended to the Multi-TransNet and further employed to estimate appropriate neuron shapes for the subdomain TransNets, greatly reducing the parameter tuning cost. Additionally, we propose a normalization approach to adaptively select the weighting parameters for the terms in the loss function. Ablation studies and extensive experiments with comparison tests on different types of elliptic interface problems with low to high contrast diffusion coefficients in two and three dimensions are carried out to numerically demonstrate the superior accuracy, efficiency, and robustness of the proposed Multi-TransNet method.

1. Introduction

Interface problems arise in numerous physical applications, such as fluid mechanics [1,2], composite materials [3,4], biological sciences [5–7] and electromagnetics [8,9], among others. In this paper, we consider the following elliptic interface problem located in an open bounded, connected domain $\Omega \in \mathbb{R}^d$:

* Corresponding author.
E-mail addresses: lutianzheng@buaa.edu.cn (T. Lu), ju@math.sc.edu (L. Ju), liyongzhu@buaa.edu.cn (L. Zhu).

https://doi.org/10.1016/j.jcp.2025.113902
Received 24 August 2024; Received in revised form 26 February 2025; Accepted 27 February 2025

Available online 5 March 2025

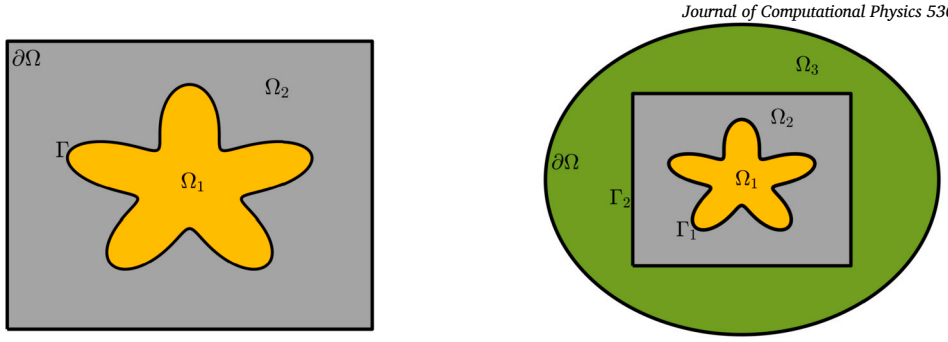

Fig. 1. Some sample domains with interfaces. Left: $K = 2$ with $\Gamma = \overline{\Omega}_1 \cap \overline{\Omega}_2$; Right: $K = 3$ with $\Gamma_1 = \overline{\Omega}_1 \cap \overline{\Omega}_2$ and $\Gamma_2 = \overline{\Omega}_2 \cap \overline{\Omega}_3$.

$$\begin{cases} \mathcal{L}(u) = f, & \boldsymbol{x} \in \cup_{k=1}^{K} \Omega_k, \\ [u] = h_1, & \boldsymbol{x} \in \Gamma_{i,j}, \quad i \neq j, \\ [\mathcal{J}(u) \cdot \boldsymbol{n}_{i,j}] = h_2, & \boldsymbol{x} \in \Gamma_{i,j}, \quad i \neq j, \\ \mathcal{B}(u) = g, & \boldsymbol{x} \in \partial\Omega, \end{cases} \tag{1}$$

where Ω is partitioned into K open connected subdomains $\{\Omega_1, \Omega_2, \ldots, \Omega_K\}$ by the closed interfaces $\{\Gamma_{i,j} = \overline{\Omega}_i \cap \overline{\Omega}_j \neq \emptyset \mid i \neq j\}$, as illustrated in Fig. 1. The operators $\mathcal{L}$ and $\mathcal{J}$ take some differential forms in subdomains and interfaces, while $\mathcal{B}$ denotes the boundary operator defined on $\partial\Omega$, which are all assumed to be linear in this work. The notation $[v]$ indicates the jump of v across the interface $\Gamma_{i,j}$, i.e., $[v] = v|_{\Omega_i} - v|_{\Omega_j}$, and the vector $\boldsymbol{n}_{i,j}$ represents the unit outward normal vector on $\Gamma_{i,j}$ from Ω_i to Ω_j.

The low regularity of solutions across interfaces, coupled with the complex geometry of these interfaces, often leads to accuracy loss when applying standard numerical methods to solve the elliptic interface problem (1). To address this, body-fitted meshes are introduced to ensure optimal or near-optimal convergence rates [10,11]. However, generating robust grids is usually time-consuming and inefficient, particularly in the case of large deformations. Consequently, researchers have turned to modifying standard numerical methods on structured grids, a line of work that can be dated back to the immersed boundary method [5,12]. Since then, a variety of interface-capturing methods have emerged, which can generally be categorized into explicit and implicit approaches according to how they handle interfaces. Explicit methods, such as the front-tracking method [13], involve explicitly tracking interfaces. While these methods offer higher accuracy in capturing interface details, they may be less efficient in dealing with changes in interface topology. In contrast, implicit approaches are more flexible in accommodating such topological changes and are often preferred in practice for complex scenarios. Notable examples include the volume of fluid method [14], which tracks the volume fraction of phases within computational cells, inherently ensuring mass conservation across the domain; the level-set method [15], which employs a signed distance function to implicitly represent the interface as the zero level-set of a scalar function, allowing for more natural handling of complex interface geometries; the immersed interface method [16,17], which modifies standard finite difference stencils near interfaces by adding correction terms to account for jumps in the solution or its derivatives, maintaining optimal accuracy up to interfaces; and the ghost fluid method [18,19], which defines ghost cells adjacent to interfaces to enforce interface conditions, enabling standard numerical schemes to be applied across discontinuities. Additionally, a variant of the ghost fluid method [20] ensures flux convergence by modifying the interface condition for fluxes in the ghost fluid method. We shall not provide an exhaustive discussion of all related traditional numerical methods here (see [21,22] for more comprehensive reviews).

In recent years, with remarkable success of neural networks across various fields [23], neural network-based numerical methods for scientific computing, especially solving partial differential equations (PDEs), have emerged. The methods such as the Deep Ritz Method (DRM) [24], the Deep Galerkin Method (DGM) [25], and the Physics-Informed Neural Networks (PINNs) [26] have gained significant attention due to their mesh-free nature. In fact, this mesh-free property is especially advantageous for addressing problems involving complex geometric domains, such as interface problems. The elliptic interface problem was tackled using a deep neural network approach based on least-square functionals of the first-order system in [27]. DRM was also employed to address elliptic interface problems with high-contrast discontinuity coefficients in [28]. Recently, localized neural network methods, including those utilizing domain decomposition techniques [29–31], have attracted increasing interest. A piecewise deep neural network method was applied to elliptic interface problems in [32], where it was numerically demonstrated that the method can accurately solve problems with complex-shaped interfaces. In [33], the Interfaced Neural Network (INN) method, which utilizes multiple neural networks, was proposed, and the multiple-gradient descent approach was extended to adaptively adjust the weighting parameters in the loss function, thereby improving the robustness of solutions for elliptic interface problems. In [34], to achieve controllable accuracy and convergence, the authors integrated an advanced level-set based finite volume numerical scheme [35] into two deep neural networks. This hybrid method was applied in parallel to solve three-dimensional elliptic interface problems involving convoluted geometries.

In practice, to achieve better approximation power, the neural networks mentioned above are typically designed with deep architectures. However, this often leads to challenging optimization problems. Given the limitations of current optimization techniques, it is often difficult to significantly reduce optimization errors, hindering the attainment of high accuracy. Based on the universal approximation theorem for two-layer (i.e., single-hidden-layer) neural networks [36], a viable approach is to employ such a shallow network structure and leverage high-order optimizers to efficiently lower optimization errors in practice. In [37,38], a high-order

full-batch optimizer was introduced to train a shallow neural network with augmented input for elliptic interface problems featuring cusps or discontinuities, leading to a significant improvement in accuracy. However, high-order optimizers are generally unstable and computationally prohibitive for large-scale training. An alternative approach is to bypass the challenging optimization problem, altogether by randomly initializing and fixing the parameters of the hidden layers, leaving only the output layer trainable. This results in a significantly simplified optimization problem, reducing it to a least squares problem that depends only on the parameters of the output layer. The linearity or nonlinearity of the problem is fully determined by the nature of PDEs. Furthermore, this approach can be efficiently solved using well-established linear or nonlinear least-squares techniques, eliminating the need of gradient-based optimizers. The local Extreme Learning Machine (locELM) [39] and the Random Feature Method (RFM) [31] have been proposed for solving PDEs. Recently, both methods have been applied to elliptic interface problems in [40] and [41], demonstrating high accuracy and efficiency, often on a par with or even surpassing traditional numerical methods. However, the fixed parameters of hidden layers in both locELM and RFM lack interpretability, which typically leads to blind parameter selection even impractical manual adjustments. The recently proposed Transferable Neural Network (TransNet) [42] addresses this issue by developing a two-layer neural network with pre-determined, uniformly distributed, shape-shared neurons in the hidden layer, offering both intuitive interpretability and excellent transferability for solving PDEs.

In this paper, we develop a novel *Multiple Transferable Neural Network* (Multi-TransNet) method for solving elliptic interface problems. This approach leverages nonoverlapping domain decomposition, tailoring multiple distinct transferable neural networks to different subdomains based on their specific features and integrating these subdomain TransNets through interface conditions incorporated into the loss function. The main contributions of our work are summarized as follows.

- We extend the property of uniform distribution of hidden-layer neurons in TransNet to the global case in Multi-TransNet, enabling the adaptive assignment of the number of neurons for each subdomain TransNet, thereby enhancing the overall transferability.
- We propose an empirical formula for TransNet to characterize the relationship between some of its key parameters, which is then extended to Multi-TransNet and employed to predict appropriate neuron shapes for subdomain TransNets, significantly reducing the parameter tuning cost.
- We propose a normalization approach to adaptively select the weighting parameters for the terms in the loss function, improving the accuracy and robustness of the Multi-TransNet method.
- We conduct extensive ablation studies to confirm the effectiveness of the proposed strategies, and perform abundant experiments with comparative tests on various types of elliptic interface problems in two and three dimensions to demonstrate the superior accuracy, efficiency, and robustness of the proposed Multi-TransNet method.

The rest of the paper is organized as follows. In Section 2, we briefly revisit the TransNet method for solving general PDEs. In Section 3, we propose an efficient empirical formula-based strategy for automatically select an appropriate shape parameter of TransNet. Subsequently, the Multi-TransNet method for elliptic interface problems is proposed and comprehensively discussed in Section 4. Extensive ablation studies and comparison tests on various types of two- and three-dimensional elliptic interface problems are presented in Section 5 to numerically demonstrate the outstanding performance of the proposed Multi-TransNet method, followed by some concluding remarks in Section 6.

2. Review of the transferable neural network method

The TransNet developed in [42] is a two-layer neural network method for solving PDEs of the following form

$$\begin{cases} \mathcal{L}(u) = f, & \boldsymbol{x} \in \Omega, \\ \mathcal{B}(u) = g, & \boldsymbol{x} \in \partial\Omega. \end{cases} \tag{2}$$

The key ingredient is the so-called neural feature space, denoted by $\mathcal{P}_{\mathrm{NN}}$, expanded by a group of neural basis functions (i.e., the neurons of the hidden layer):

$$\mathcal{P}_{\mathrm{NN}} = \mathrm{span}\left\{1, \sigma\left(\boldsymbol{w}_1^{\top}\boldsymbol{x} + b_1\right), \ldots, \sigma\left(\boldsymbol{w}_M^{\top}\boldsymbol{x} + b_M\right)\right\}, \tag{3}$$

where $\sigma(\cdot)$ is the activation function, and $\boldsymbol{w}_m \in \mathbb{R}^d, b_m \in \mathbb{R}$ is the weight and bias of the m-th hidden-layer neuron, respectively. The TransNet solution $u_{\mathrm{NN}} \in \mathcal{P}_{\mathrm{NN}}$ is represented by a linear combination of the neural basis functions

$$u_{\mathrm{NN}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m \sigma\left(\boldsymbol{w}_m^{\top}\boldsymbol{x} + b_m\right) + \alpha_0, \tag{4}$$

where $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_M$ are the parameters of the output layer in TransNet (one may directly take $\alpha_0 = 0$ in practice). *The main idea of TransNet is that the neurons of its hidden layer will be pre-determined in a special sense of uniform distribution from the pure approximation point of view before applied to the solution of PDEs*. Below we briefly review the process of the TransNet method for solving the PDE problem (2).

2.1. Geometrization of the neural feature space

Inspired by activation patterns of ReLU networks, the hidden-layer neurons of TransNet are geometrized based on a re-parameterization of the neural feature space $\mathcal{P}_{\mathrm{NN}}$ without using any PDE information, and then are naturally connected with the computational domain Ω. Specifically, a hidden-layer neuron represented by a piecewise-linear function in a ReLU network can be viewed as a *partition hyperplane*, i.e.,

$$\boldsymbol{w}_m^{\top}\boldsymbol{x} + b_m = 0, \tag{5}$$

in a geometric space. It can be rewritten into

$$\gamma_m \boldsymbol{a}_m^{\top}\left(\boldsymbol{x} + r_m \boldsymbol{a}_m\right) = 0, \tag{6}$$

similar to the point-normal form of a hyperplane equation in $\mathbb{R}^d$, in which both the unit normal vector $\boldsymbol{a}_m \in \mathbb{R}^d$ of the partition hyperplane (5) and the scalar $r_m \in \mathbb{R}$ representing the distance between the origin and the partition hyperplane (5) are collectively referred to as the *location parameter*, while the scalar $\gamma_m \in \mathbb{R}_+$ is named as the *shape parameter* owing to its association with the geometric shape characteristic of $\sigma\left(\gamma_m\left(\boldsymbol{a}_m^{\top}\boldsymbol{x} + r_m\right)\right)$ (see Figure 1 in [42] for their visualization). Evidently, the relationship between the original parameters $\boldsymbol{w}_m, b_m$ and the current ones $\boldsymbol{a}_m, r_m, \gamma_m$ is given by

$$\boldsymbol{w}_m = \gamma_m \boldsymbol{a}_m, \quad b_m = \gamma_m r_m. \tag{7}$$

Therefore, the TransNet solution (4) is rewritten as

$$u_{\mathrm{NN}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m \sigma\left(\gamma_m\left(\boldsymbol{a}_m^{\top}\boldsymbol{x} + r_m\right)\right) + \alpha_0. \tag{8}$$

The next step of TransNet is to obtain the location parameter $(\boldsymbol{a}_m, r_m)$ and the shape parameter γ_m of the hidden-layer neuron, i.e., the *pre-training*.

2.2. Generating the hidden-layer neuron location — uniform neuron distribution

In a ReLU network, a hidden-layer neuron can be regarded as a partition hyperplane (5) to some extent, and a region intersected by multiple such partition hyperplanes can define a linear piece. Obviously, adding the number of linear pieces is the most straightforward way to enhance the approximation power of the network for solving PDEs. Furthermore, intuitively, the location of linear pieces should have a significant influence on the generalization of the network for solving PDEs defined in different domains. For these reasons, a concept of *uniform neuron distribution* is designed and related construction algorithm is rigorously proven in [42].

Theorem 1 *(Uniform neuron distribution in the unit ball $B_1(\mathbf{0})$). Given a set of M partition hyperplanes of $\mathbb{R}^d$ defined by*

$$\boldsymbol{a}_m^{\top}\boldsymbol{x} + r_m = 0, \quad m = 1, 2, \dots M.$$

If $\{\boldsymbol{a}_m\}_{m=1}^{M}$ are i.i.d. and uniformly distributed on the d-dimensional unit sphere, and $\{r_m\}_{m=1}^{M}$ are i.i.d. and uniformly distributed in $[0,1]$, then for a fixed $\tau \in (0,1)$,

$$\mathbb{E}\left[D_M^{\tau}(\boldsymbol{x})\right] = \tau, \quad \forall\, \|\boldsymbol{x}\|_2 \leqslant 1 - \tau,$$

where $D_M^{\tau}(\boldsymbol{x})$ is the density function of the neurons defined by

$$D_M^{\tau}(\boldsymbol{x}) = \frac{1}{M}\sum_{m=1}^{M} \chi_{\{d_m(\boldsymbol{x})<\tau\}}(\boldsymbol{x}),$$

with $\chi_{\{d_m(\boldsymbol{x})<\tau\}}(\boldsymbol{x})$ denoting the indicator function whether the distance between $\boldsymbol{x}$ and the m-th partition hyperplane is smaller than τ.

Note that in practice, $\{\boldsymbol{a}_m\}_{m=1}^{M}$ satisfying the requirement in Theorem 1 can be obtained by sampling from the d-dimensional standard Gaussian distribution in the Cartesian coordinate system and then normalizing the samples to unit vectors. Theorem 1 limits the domain of computation to a unit ball, but one can naturally generalize it to the case of a ball $\boldsymbol{B}_R(\boldsymbol{x}_c)$ centered at the point $\boldsymbol{x}_c \in \mathbb{R}^d$ with a radius $R \in \mathbb{R}_+$ with an affine (scaling and translation) transformation, and thus obtain the following result.

Theorem 2 *(Uniform neuron distribution in the ball $B_R(\boldsymbol{x}_c)$). Given a set of M partition hyperplanes of $\mathbb{R}^d$ defined by*

$$\boldsymbol{a}_m^{\top}(\boldsymbol{x} - \boldsymbol{x}_c) + r_m = 0, \quad m = 1, 2, \dots M.$$

If $\{\boldsymbol{a}_m\}_{m=1}^{M}$ are i.i.d. and uniformly distributed on the d-dimensional unit sphere, and $\{r_m\}_{m=1}^{M}$ are i.i.d. and uniformly distributed in $[0,R]$, then for a fixed $\tau \in (0,R)$,

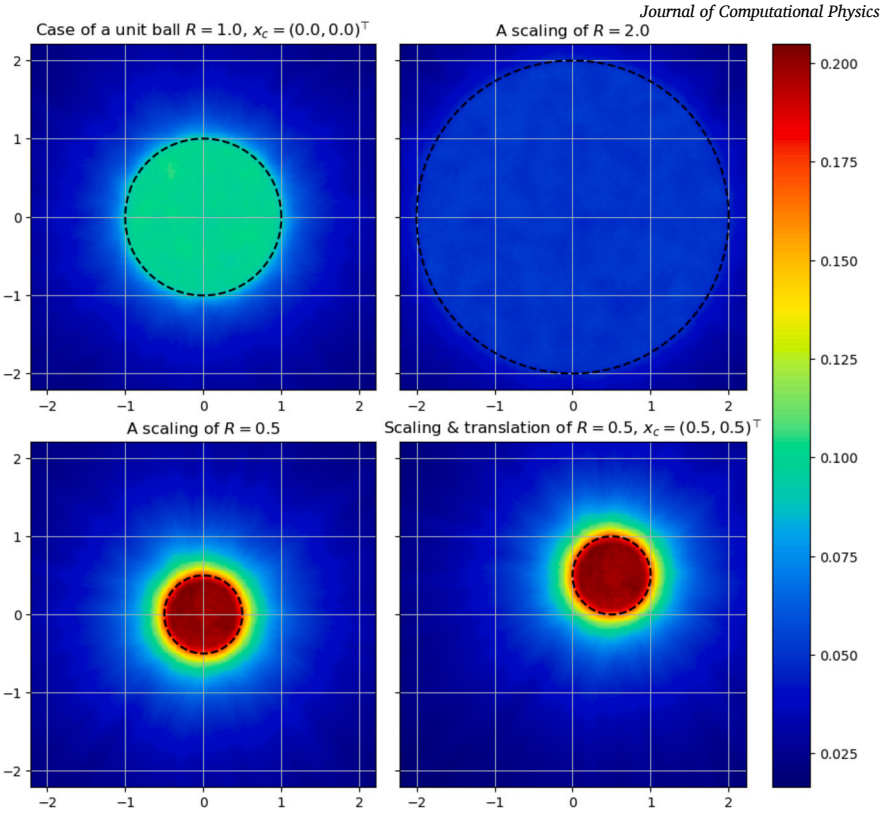

Fig. 2. Illustration of the density distribution of the partition hyperplanes of hidden-layer neurons from TransNet with different $\boldsymbol{x}_c$ and R. Here $M = 30,000$ and $\tau = 0.1$. Top-left to top-right: with a scaling of factor of 2.0, top-left to bottom-left: with a scaling of factor of 0.5, and top-left to bottom-right: with a scaling of factor of 0.5 and a translation of the center $(0.5, 0.5)$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$\mathbb{E}\left[D_M^{\tau}(\boldsymbol{x})\right] = \frac{\tau}{R}, \quad \forall \|\boldsymbol{x} - \boldsymbol{x}_c\|_2 \leqslant R - \tau. \tag{9}$$

For the sake of geometric intuition, we illustrate the uniform neuron distribution with some examples in Fig. 2. It is clearly observed that the scaling of the hidden-layer neurons is achieved from the case of $R = 1.0$ and $\boldsymbol{x}_c = (0,0)^\top$ (top-left) to the case of $R = 2$ (top-right) and the case of $R = 0.5$ (bottom-left), and the corresponding density of hidden-layer neurons doubles or halves. The hidden-layer neurons of bottom-left further undergo a translation from the center $(0,0)$ to another center $(0.5, 0.5)$ (bottom-right), and the density remains unchanged.

Remark 1 *(The selection of parameters $\boldsymbol{x}_c$ and R for a general domain Ω).* For a given domain Ω, the center of the ball $\boldsymbol{x}_c$ should be selected to be approximately the center of the domain Ω and the radius of the ball R should be chosen to make the ball $B_R(\boldsymbol{x}_c)$ slightly over-cover the domain Ω, to enhance the efficacy and density of the hidden-layer neurons within the domain.

2.3. Tuning the hidden-layer neuron shape and solving the PDE

In order to strive for simplicity and generalization, all hidden-layer neurons of TransNet are assumed to share the same shape parameter in [42], i.e., $\gamma_m = \gamma$ for $m = 1, 2, \ldots, M$. The Gaussian Random Fields (GRFs) are introduced to generate auxiliary functions for pre-tuning γ based on the function approximation ability, and the grid search is used for finding an optimal value of γ which minimizes the mean approximation errors over the whole set of auxiliary functions. The entire tuning process is *offline* and only need to solve some linear least squares problems. The neural feature space (3) is then completely constructed without using any PDE information, and the TransNet solution (8) is reduced to

$$u_{\mathrm{NN}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m \sigma\left(\gamma\left(\boldsymbol{a}_m^\top\left(\boldsymbol{x} - \boldsymbol{x}_c\right) + r_m\right)\right) + \alpha_0. \tag{10}$$

The final step of TransNet for solving the problem (2) is to obtain the parameters $\boldsymbol{\alpha} = \{\alpha_0, \alpha_1, \ldots, \alpha_M\}$ of the output layer. This can be done by substituting the TransNet solution (10) into (2) and minimizing the following physics-informed loss function

$$\mathrm{Loss}_{\mathrm{TN}}(\boldsymbol{\alpha}) = \lambda_L \|\mathcal{L}(u_{\mathrm{NN}}(\boldsymbol{x})) - f(\boldsymbol{x})\|_2^2 + \lambda_B \|\mathcal{B}(u_{\mathrm{NN}}(\boldsymbol{x})) - g(\boldsymbol{x})\|_2^2 \tag{11}$$

over a set of training/collocation points, where λ_L and λ_B are some positive weighting parameters. This is essentially solving a linear (or nonlinear if the operators $\mathcal{L}$ and/or $\mathcal{B}$ in (2) are nonlinear) least squares problem, which can be efficiently done with preferred QR factorization related techniques.

3. An efficient empirical formula-based prediction strategy for the shape parameter of TransNet

As stated in [42], the tuning process for the shape parameter reviewed in Section 2.3 does not use any information from PDE problems and is mainly targeted at providing some reasonable values for the shape parameter γ in practice and enhancing the transferability of the TransNet across various PDEs with different domains and boundary conditions. Additionally, the use of GRFs for generating auxiliary functions also requires the selection of a new parameter, the correlation length of GRF, an appropriate choice of which is usually related to the variations of the PDE solutions to be solved. Furthermore, the optimal value of γ for a TransNet also varies along with the number of hidden-layer neurons M even for the same PDE problem. Hence, such tuning approach could fail to work and inevitably sacrifice much accuracy of the TransNet method in some situations.

For a specific PDE problem (2), assume that the ball $B_R(\boldsymbol{x}_c)$ has been determined for the domain Ω and the number of hidden-layer neurons M is given. The location parameters $\{(\boldsymbol{a}_m, r_m)\}_{m=1}^M$ then can be easily generated according to the process described in Theorem 2. Notice that the physics-informed loss function (11) is a natural error indicator for accuracy of the TransNet solution (8), so we can define the following posterior error indicator function for TransNet with respect to a given shape parameter γ:

$$\eta(\gamma) = \min_{\boldsymbol{\alpha}} \mathrm{Loss}_{\mathrm{TN}}(\boldsymbol{\alpha}).$$

However, the derivative of $\eta(\gamma)$ is hard to derive, and consequently a simple but effective way is to utilize a *gradient-free* line search technique to find optimal shape parameter, i.e., solve the minimization problem

$$\min_{\gamma} \eta(\gamma). \tag{12}$$

Here we take the popular *golden-section search* algorithm and the details are described in Algorithm 1. Note that each evaluation of the objective function $\eta(\gamma)$ costs one TransNet solving. We will call this approach *the training loss-based optimization strategy* for searching optimal shape parameter of the TransNet.

It is worth noting that when the number of hidden-layer neurons M is small, the proposed training loss-based optimization strategy is efficient due to the low computational cost for a small-scale TransNet. However, when M gets larger and larger, the cost of each TransNet solving increase significantly and this optimization strategy could become computationally expensive and even not acceptable in practice, thus a more efficient selection strategy is still desired to resolve this issue.

Algorithm 1: The golden-section search algorithm for the shape parameter of TransNet.

Input: A TransNet with the number of hidden-layer neurons M and the location parameters $\{(\boldsymbol{a}_m, r_m)\}_{m=1}^M$ generated for the ball $B_R(\boldsymbol{x}_c)$, the search interval $[a, b]$, and the number of iterations Itr

Output: Optimal shape parameter γ^{opt}

Set $\omega = (\sqrt{5}-1)/2$, $\gamma^{(1)} = b - \omega(b-a)$, $\gamma^{(2)} = a + \omega(b-a)$;
Solve (2) using the TransNet with $\gamma^{(1)}$ and $\gamma^{(2)}$ as the shape parameter and obtain the loss function value $res^{(1)}$ and $res^{(2)}$, respectively ;
for $i = 1 : Itr$ **do**
  if $res^{(1)} \leqslant res^{(2)}$ **then**
    $b := \gamma^{(2)}$; $\gamma^{(2)} := \gamma^{(1)}$; $\gamma^{(1)} := b - \omega(b-a)$; $res^{(2)} := res^{(1)}$;
    Solve (2) using the TransNet with the shape parameter $\gamma^{(1)}$ to obtain a new loss value $res^{(1)}$;
  else
    $a := \gamma^{(1)}$; $\gamma^{(1)} := \gamma^{(2)}$; $\gamma^{(2)} := a + \omega(b-a)$; $res^{(1)} := res^{(2)}$;
    Solve (2) using the TransNet with the shape parameter $\gamma^{(2)}$ to obtain a new loss value $res^{(2)}$;
  end
end
if $res^{(1)} \leqslant res^{(2)}$ **then**
  $\gamma^{\mathrm{opt}} := \gamma^{(1)}$
else
  $\gamma^{\mathrm{opt}} := \gamma^{(2)}$
end

We observe from the results of [42] that the optimal shape parameter γ^{opt} is positively related to both the number of hidden-layer neurons M and the variation speed of the solution. On the other hand, (7) and Theorem 2 tell us that the product of γ and R determines the value range of b_m, so there also exists a certain inverse relationship between γ and R. Based on the above analysis, we propose an empirical formula (13) for characterizing the relation between these parameters of TransNet as follows:

$$\gamma \approx C \frac{M^{1/d}}{R}, \tag{13}$$

where C is called the *empirical constant* which depends on the settings of the target PDE problem, and d is the problem dimension.

By combining the training loss-based optimization strategy and the empirical formula (13), we propose *an empirical formula-based prediction strategy* for estimating appropriate shape parameter of the TransNet. The strategy consists of a preprocessing step and a prediction step as follows:

- **Preprocessing**: Select a small number of hidden-layer neurons M_0 and use the training loss-based optimization strategy (e.g., Algorithm 1) to find an optimal shape parameter γ_0^{opt} for the current TransNet, and consequently, set the empirical constant $C := \frac{\gamma_0^{\text{opt}} R}{(M_0)^{1/d}}$ in the empirical formula (13).
- **Prediction**: Given any new (usually larger) value of M to be used for the TransNet, directly predict an appropriate shape parameter γ_* according to the empirical formula (13), that is, $\gamma_* := C\frac{M^{1/d}}{R}$.

Note that the prediction step is completely explicit (does not involve any training or optimization) and the computational cost for the preprocessing step is low since M_0 is small, and thus this strategy is very efficient in practice.

4. A multiple transferable neural network method for elliptic interface problems

It is well known that the activation function plays a pivotal role in the expressive power of the neural network and it is generally continuous and even differentiable. However, the solution to interface problems often exhibits nonsmoothness or even discontinuities, which incurs poor performance of neural network methods. Specifically, for the elliptic interface problem (1) with K subdomains, the solution in the whole domain is divided into K parts located in different subdomains as illustrated in Fig. 1, in each of which the solution is usually smooth, but has cusps or jumps at the interfaces. In order to conquer such a problem, a natural idea is to respectively employ a neural network to approximate the smooth solution over each subdomain and then to unite them through the interface conditions on interfaces as done by many existing numerical methods. Therefore, we integrate multiple *tailored* TransNets related to subdomains using the *nonoverlapping domain decomposition* approach to develop a novel multiple transferable neural network method, abbreviated as Multi-TransNet, for solving the elliptic interface problem (1). Fig. 3 illustrates the solution process of the proposed Multi-TransNet method when $K = 2$.

4.1. Nonoverlapping domain decomposition and integrated subdomain TransNets

Without loss of generality, here we take the common case of $K = 2$ subdomains (Ω_1 and Ω_2) and one interface Γ (see Fig. 1-left) for an illustration of the proposed Multi-TransNet method for solving the elliptic interface problem (1). To be detailed, we employ two TransNets u_1^{NN} and u_2^{NN} with respective M_1 and M_2 hidden-layer neurons to approximate the solution in Ω_1 and Ω_2, respectively, and then the Multi-TransNet solution with totally $M = M_1 + M_2$ hidden-layer neurons can be represented by

$$u^{\text{NN}} = u_1^{\text{NN}}\,\chi_{\Omega_1} + u_2^{\text{NN}}\,\chi_{\Omega_2}, \tag{14}$$

where $\left\{\chi_{\Omega_k}\right\}_{k=1}^{2}$ are the indicator functions and

$$u_k^{\text{NN}} = \boldsymbol{\alpha}_k^{\top}\boldsymbol{\phi}_k, \quad k = 1, 2, \tag{15}$$

with $\boldsymbol{\alpha}_k = \left(\alpha_0^{(k)}, \alpha_1^{(k)}, \ldots, \alpha_{M_k}^{(k)}\right)^{\top}$, $\boldsymbol{\phi}_k = \left(\phi_0^{(k)}, \phi_1^{(k)}, \ldots, \phi_{M_k}^{(k)}\right)^{\top}$, and the neural basis functions

$$\phi_0^{(k)} = 1, \quad \phi_m^{(k)}(\boldsymbol{x}) = \sigma\left(\gamma_k\left(\left(\boldsymbol{x} - \boldsymbol{x}_c^{(k)}\right)^{\top}\boldsymbol{a}_m^{(k)} + r_m^{(k)}\right)\right), \quad m = 1, 2, \ldots, M_k.$$

The corresponding loss function of our Multi-TransNet for solving (1) is then designed to be

$$\begin{aligned}\text{Loss}_{\text{MT}}(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2) = &\left\|\lambda_1\left(\mathcal{L}\left(u^{\text{NN}}\right) - f\right)\right\|_{\Omega_1,2}^2 + \left\|\lambda_2\left(\mathcal{L}\left(u^{\text{NN}}\right) - f\right)\right\|_{\Omega_2,2}^2 + \left\|\lambda_g\left(\mathcal{B}\left(u^{\text{NN}}\right) - g\right)\right\|_{\partial\Omega,2}^2 \\ &+ \left\|\lambda_{h,1}\left(\left[u^{\text{NN}}\right] - h_1\right)\right\|_{\Gamma,2}^2 + \left\|\lambda_{h,2}\left(\left[\mathcal{J}\left(u^{\text{NN}}\right)\cdot\boldsymbol{n}\right] - h_2\right)\right\|_{\Gamma,2}^2,\end{aligned} \tag{16}$$

where λ_1, λ_2, λ_g, $\lambda_{h,1}$, $\lambda_{h,2}$ are some positive weighting parameters.

In practical implementation, assume that N_1, N_2, N_g and N_Γ training/collocation points have been sampled in Ω_1, Ω_2, $\partial\Omega$ and Γ, respectively, which are arranged as follows:

$$\boldsymbol{X}_1 = \begin{pmatrix}\left(\boldsymbol{x}_1^{(1)}\right)^{\top}\\ \left(\boldsymbol{x}_2^{(1)}\right)^{\top}\\ \vdots\\ \left(\boldsymbol{x}_{N_1}^{(1)}\right)^{\top}\end{pmatrix}, \quad \boldsymbol{X}_2 = \begin{pmatrix}\left(\boldsymbol{x}_1^{(2)}\right)^{\top}\\ \left(\boldsymbol{x}_2^{(2)}\right)^{\top}\\ \vdots\\ \left(\boldsymbol{x}_{N_2}^{(2)}\right)^{\top}\end{pmatrix}, \quad \boldsymbol{X}_g = \begin{pmatrix}\boldsymbol{x}_{g,1}^{\top}\\ \boldsymbol{x}_{g,2}^{\top}\\ \vdots\\ \boldsymbol{x}_{g,N_g}^{\top}\end{pmatrix}, \quad \boldsymbol{X}_\Gamma = \begin{pmatrix}\boldsymbol{x}_{\Gamma,1}^{\top}\\ \boldsymbol{x}_{\Gamma,2}^{\top}\\ \vdots\\ \boldsymbol{x}_{\Gamma,N_\Gamma}^{\top}\end{pmatrix}. \tag{17}$$

Next, we substitute the Multi-TransNet solution (14) into the elliptic interface problem (1) and evaluate them at the above training/collocation points, and then minimize the squared residual (i.e., the loss function (16) in the discrete sense), i.e.,

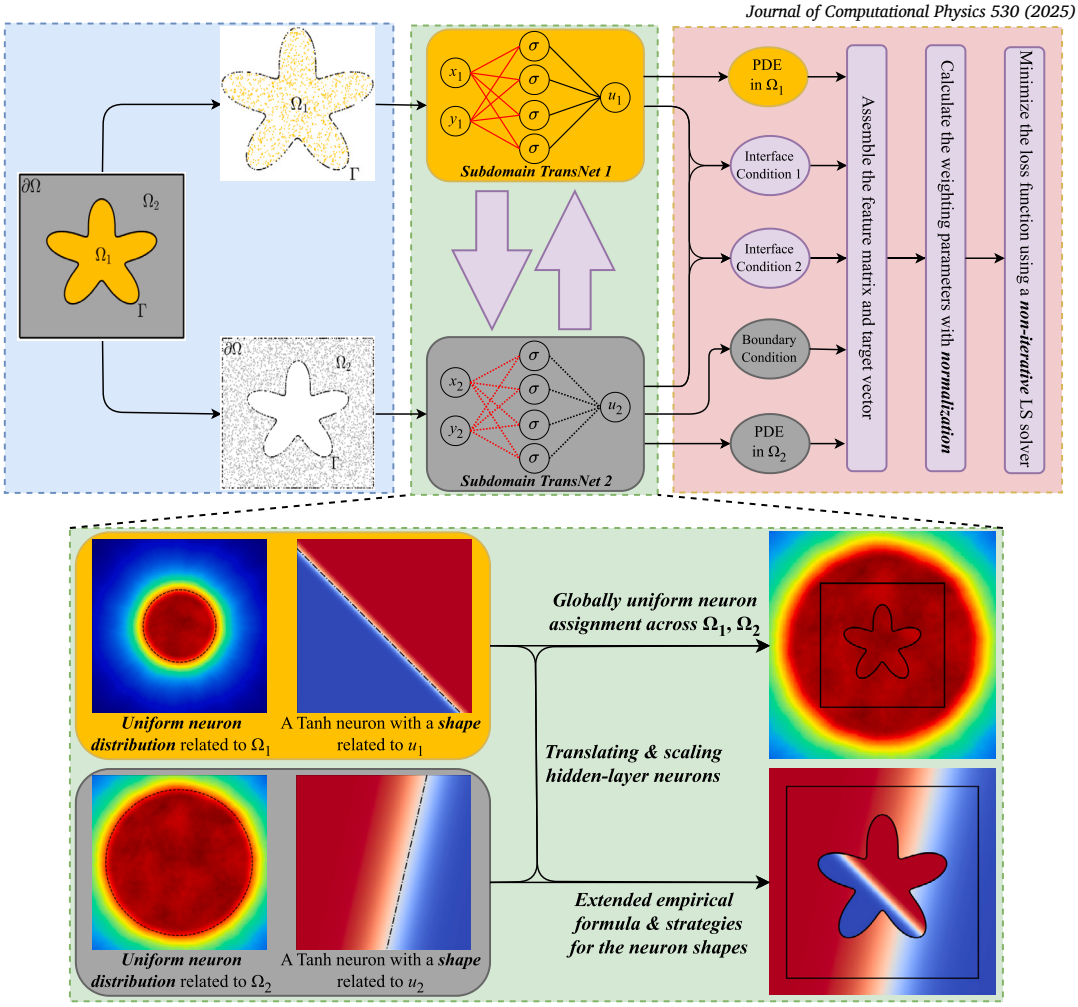

Fig. 3. Illustration of the solution process of the proposed Multi-TransNet method for the elliptic interface problem (1) with $K = 2$.

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{F}\boldsymbol{\alpha} - \boldsymbol{T}\|_2^2, \tag{18}$$

where the feature matrix $\boldsymbol{F}$, the target vector $\boldsymbol{T}$ and the parameters $\boldsymbol{\alpha}$ of the output layer are respectively assembled in the following manner:

$$\boldsymbol{F} = \left(\begin{array}{c|c} \lambda_1 \mathcal{L}\left(\boldsymbol{\phi}_1^{\top}(\boldsymbol{X}_1)\right) & \boldsymbol{O} \\ \boldsymbol{O} & \lambda_2 \mathcal{L}\left(\boldsymbol{\phi}_2^{\top}(\boldsymbol{X}_2)\right) \\ \boldsymbol{O} & \lambda_g \mathcal{B}\left(\boldsymbol{\phi}_2^{\top}(\boldsymbol{X}_g)\right) \\ \lambda_{h,1}\boldsymbol{\phi}_1^{\top}(\boldsymbol{X}_\Gamma) & -\lambda_{h,1}\boldsymbol{\phi}_2^{\top}(\boldsymbol{X}_\Gamma) \\ \lambda_{h,2}\mathcal{J}\left(\boldsymbol{\phi}_1^{\top}(\boldsymbol{X}_\Gamma)\right)\cdot\boldsymbol{n} & -\lambda_{h,2}\mathcal{J}\left(\boldsymbol{\phi}_2^{\top}(\boldsymbol{X}_\Gamma)\right)\cdot\boldsymbol{n} \end{array}\right), \quad \boldsymbol{T} = \begin{pmatrix} \lambda_1 f(\boldsymbol{X}_1) \\ \lambda_2 f(\boldsymbol{X}_2) \\ \lambda_g g(\boldsymbol{X}_g) \\ \lambda_{h,1} h_1(\boldsymbol{X}_\Gamma) \\ \lambda_{h,2} h_2(\boldsymbol{X}_\Gamma) \end{pmatrix}, \quad \boldsymbol{\alpha} = \begin{pmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \end{pmatrix}. \tag{19}$$

It is clearly $\boldsymbol{F} \in \mathbb{R}^{(N_1+N_2+N_g+2N_\Gamma)\times(M+2)}$, $\boldsymbol{T} \in \mathbb{R}^{N_1+N_2+N_g+2N_\Gamma}$ and $\boldsymbol{\alpha} \in \mathbb{R}^{M+2}$, and the minimization problem (18) is in fact a linear least squares problem, thus $\boldsymbol{F}$ is also called the least squares coefficient matrix.

Remark 2. To briefly demonstrate the advantage of the proposed Multi-TransNet method over the TransNet method, we consider a 1D elliptic interface problem as follows:

$$\begin{cases} -\left(\beta u'\right)' = f, & x \in \Omega_1 \cup \Omega_2 = (0, 1/4) \cup (1/4, 1), \\ [u] = h_1, & x \in \Gamma = \{1/4\}, \\ \left[\beta u'\right] = h_2, & x \in \Gamma = \{1/4\}, \\ u = g, & x \in \partial\Omega = \{0, 1\}, \end{cases} \tag{20}$$

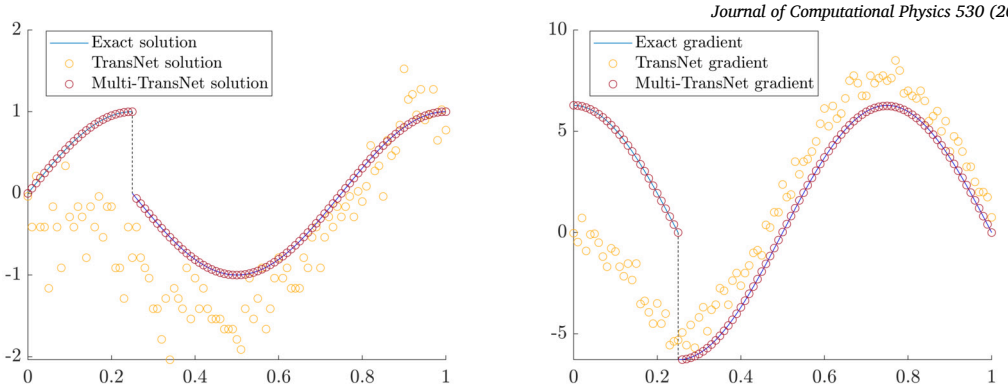

Fig. 4. Comparisons of the numerical solutions (left) and their gradients (right) produced by one TransNet with 100 hidden-layer neurons and the proposed Multi-TransNet with 5 hidden-layer neurons for each of the two subdomain TransNets for the 1D elliptic interface problem (20).

where $\beta = 1$ for $x \in (0, 1/4)$ and $\beta = 10$ for $x \in (1/4, 1)$. The exact solution is set to be

$$u(x) = \begin{cases} \sin(2\pi x), & x \in [0, 1/4], \\ \cos(2\pi x), & x \in (1/4, 1], \end{cases} \tag{21}$$

and f, h_1, h_2, g in (20) can be obtained by substituting (21) into (20). Fig. 4 illustrates the comparisons of the numerical solutions and their gradients obtained by using one TransNet with 100 hidden-layer neurons and the proposed Multi-TransNet with 5 hidden-layer neurons for each of the two subdomain TransNets for the 1D elliptic interface problem (20). It is easily observed that the TransNet method fails while the Multi-TransNet method catches the discontinuous solution and gradient accurately.

The extension of the Multi-TransNet method to the case of $K > 2$ subdomains $\{\Omega_k\}_{k=1}^K$ with multiple interfaces $\{\Gamma_{i,j}\}$ is straightforward. We assign each subdomain Ω_k a TransNet u_k^{NN} with M_k hidden-layer neurons to form a total of K subdomain TransNets $\{u_k^{\mathrm{NN}}\}_{k=1}^K$ such that the Multi-TransNet solution is given by

$$\begin{aligned} u^{\mathrm{NN}} &= \sum_{k=1}^K u_k^{\mathrm{NN}} \, \chi_{\Omega_k} = \sum_{k=1}^K \boldsymbol{\alpha}_k^\top \boldsymbol{\phi}_k \, \chi_{\Omega_k} \\ &= \sum_{k=1}^K \left(\alpha_0^{(k)} + \sum_{m=1}^{M_k} \alpha_m^{(k)} \sigma\left(\gamma_k \left(\left(\boldsymbol{x} - \boldsymbol{x}_c^{(k)} \right)^\top \boldsymbol{a}_m^{(k)} + r_m^{(k)} \right) \right) \right) \chi_{\Omega_k}. \end{aligned} \tag{22}$$

The way of constructing the loss function, assembling the feature matrix, the target vector and the parameters of the output layer is similar to (16)-(19).

4.2. Globally uniform neuron distribution across subdomains

Rather than arbitrarily assigning or equally distributing the number of hidden-layer neurons among all subdomains, we consider the globally uniform neuron distribution of Multi-TransNet for the elliptic interface problem (1) with K subdomains. At first, from the view of the neuron location, the hidden-layer neurons for each of the K subdomain TransNets should be respectively translated to the approximate center $\boldsymbol{x}_c^{(k)}$ of the corresponding subdomain Ω_k, and also be respectively scaled to enable the ball $B_{R_k}(\boldsymbol{x}_c^{(k)})$ to slightly over-cover the subdomain Ω_k, as emphasized in Remark 1. Since the uniform distribution of hidden-layer neurons can bring better transferability and accuracy for TransNet for solving general PDE problems as demonstrated in [42], a natural question is how to extend the uniform distribution feature of hidden-layer neurons on each subdomain to the *globally* uniform distribution across the whole domain for the Multi-TransNet.

Let us rewrite the main result (9) in Theorem 2 into the following form:

$$\mathbb{E}\left[\sum_{m=1}^M \chi_{\{d_m(\boldsymbol{x}) < \tau\}}(\boldsymbol{x}) \right] = \frac{M}{R}\tau, \quad \forall \, \|\boldsymbol{x} - \boldsymbol{x}_c\|_2 \leqslant R - \tau, \tag{23}$$

which implies that the expected number of hidden-layer neurons whose corresponding partition hyperplanes intersect the neighborhood $B_\tau(\boldsymbol{x})$ is proportional to M/R. It is natural to assume that the same neighborhood size τ is used for defining the acting range of hidden-layer neurons in each subdomain TransNet of the Multi-TransNet. As a result, in order to obtain the globally uniform distribution of hidden-layer neurons, we need to make the ratio M_k/R_k as equal as possible (note M_k is always an integer), i.e.,

$$\frac{M_1}{R_1} \approx \frac{M_2}{R_2} \approx \cdots \approx \frac{M_K}{R_K}. \tag{24}$$

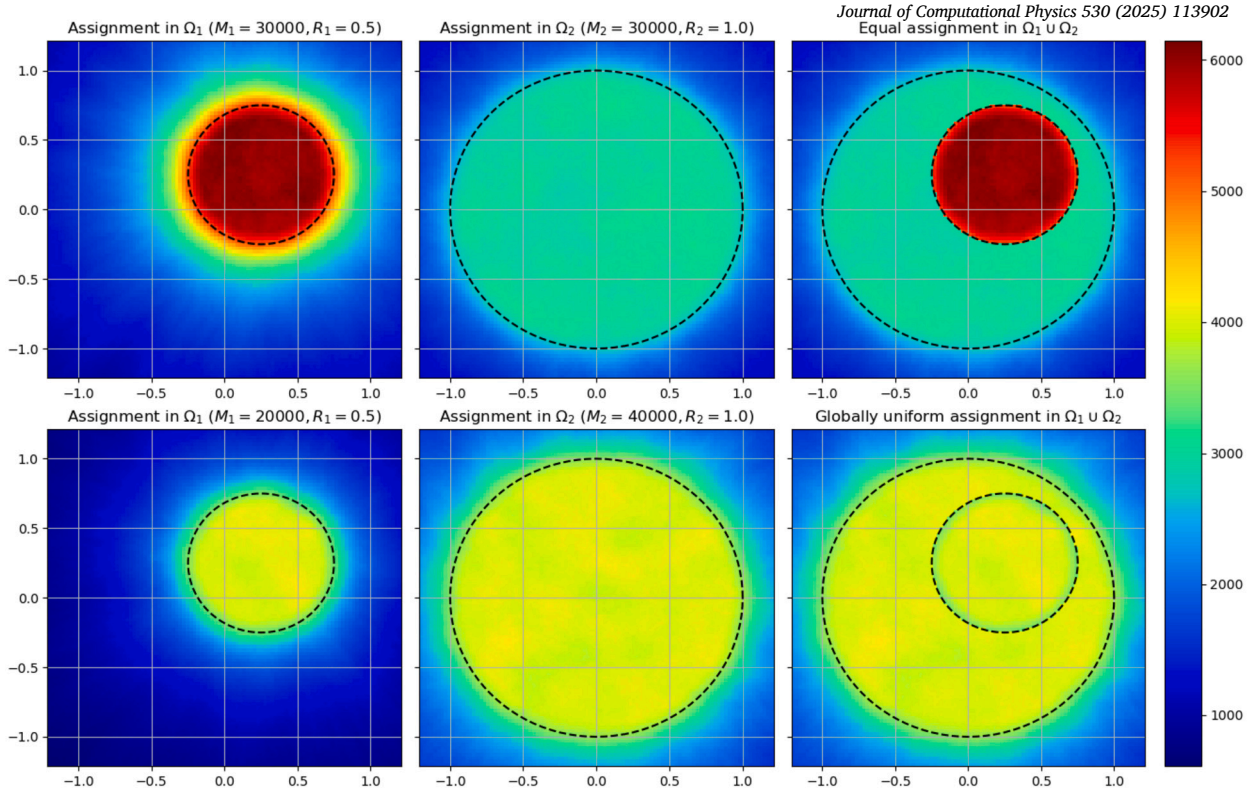

Fig. 5. Comparison of the number distribution of hidden-layer neurons of the Multi-TransNet with two subdomain TransNets with the subdomains $\Omega_1 = B_{0.5}(0.25, 0.25)$ and $\Omega_2 = B_1(0,0) \setminus B_{0.5}(0.25, 0.25)$. Here $M_1 + M_2$ and τ are fixed to 60000 and 0.1, respectively. Top row: $M_1 = M_2 = 30000$; Bottom row: $M_1 = 20000$ and $M_2 = 40000$ thus $M_1/R_1 = M_2/R_2$.

Fig. 5 illustrates comparison of the number distribution of hidden-layer neurons of the Multi-TransNet with two subdomain TransNets under the setting of the equal assignment (i.e., $M_1 = M_2$) and $M_1/R_1 = M_2/R_2$, where $\Omega_1 = B_{0.5}(0.25, 0.25)$ and $\Omega_2 = B_1(0,0) \setminus B_{0.5}(0.25, 0.25)$. It is easy to see that the setting of $M_1/R_1 = M_2/R_2$ indeed ensures the globally uniform distribution of hidden-layer neurons while the setting of the equal assignment only achieves the locally uniform distribution of those and enables more hidden-layer neurons to gather in the interior circular subdomain Ω_1. Hence, for achieving globally uniform neuron distribution, we propose to use the relation (24) to assign the number of neurons for each subdomain TransNet, which is equivalent to

$$M_k \approx M \frac{R_k}{\sum_{k=1}^{K} R_k}, \tag{25}$$

where $M = \sum_{k=1}^{K} M_k$ is the total number of hidden-layer neurons of the Multi-TransNet.

4.3. Extended empirical formula and strategies for the hidden-layer neuron shapes of Multi-TransNet

For simplicity of illustration, we first pay attention to the shape parameters of a Multi-TransNet with two subdomain TransNets. As introduced in Subsection 2.3, the shape parameter determines the variation speed of the neural basis function and affects the approximation performance of TransNet, so does Multi-TransNet. The solutions in the subdomains Ω_1 and Ω_2 could be significantly distinct, thus we need to provide respective shape parameter for each of the two subdomain TransNets (i.e., γ_1 and γ_2) and a corresponding efficient tuning strategy for optimal ones is again needed. Similar to the TransNet case, we define a posterior error indicator function for the Multi-TransNet with respect to given shape parameters (γ_1, γ_2) based on the loss function (16) as follows:

$$\eta(\gamma_1, \gamma_2) = \min_{(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2)} \mathrm{Loss}_{\mathrm{MT}}(\boldsymbol{\alpha}).$$

Furthermore, we propose to *bridge* the shape parameters of the two subdomain TransNets by extending the empirical formula (13) introduced for the TransNet in Section 3 into the Multi-TransNet case. Specifically, we assume that the two subdomain TransNets with the respective parameters, (M_1, R_1) and (M_2, R_2), share the same empirical constant C in (13), i.e.,

$$\gamma_1 \approx C \frac{M_1^{1/d}}{R_1}, \quad \gamma_2 \approx C \frac{M_2^{1/d}}{R_2}, \tag{26}$$

then consequently the following relation holds

$$\gamma_2 \approx \gamma_1 \frac{R_1}{R_2}\left(\frac{M_2}{M_1}\right)^{1/d}. \tag{27}$$

Therefore, the bivariate optimization problem for searching optimal shape parameters (γ_1, γ_2) of the Multi-TransNet can be converted into a univariate one with respect to γ_1 (or γ_2) only, i.e.,

$$\min_{\gamma_1} \eta\left(\gamma_1, \gamma_1 \frac{R_1}{R_2}\left(\frac{M_2}{M_1}\right)^{1/d}\right). \tag{28}$$

Then the training loss-based optimization strategy also can be slightly modified and used to find optimal γ_1 for the problem (28), which then can be applied to (26) to compute the needed empirical constant C. Consequently, the efficient empirical formula-based prediction strategy presented in Section 3 also can be similarly generalized to the case of Multi-TransNet for selecting appropriate values of the shape parameters γ_1 and γ_2.

In the case of K subdomain TransNets with the parameters $\{M_k\}_{k=1}^K$ and $\{R_k\}_{k=1}^K$, for the set of shape parameters $\{\gamma_k\}_{k=1}^K$ of the corresponding Multi-TransNet, we have under the same principle:

$$\gamma_k \approx C\frac{M_k^{1/d}}{R_k}, \quad k = 1, 2, \cdots, K, \tag{29}$$

and thus

$$\gamma_k \approx \gamma_1 \frac{R_1}{R_k}\left(\frac{M_k}{M_1}\right)^{1/d}, \quad k = 2, 3, \ldots, K. \tag{30}$$

Thus the two selection strategies discussed above can be straightforwardly extended.

Remark 3. In particular, when $M_1 \approx M_2 \approx \cdots \approx M_k$, we have

$$\gamma_k \approx \gamma_1 \frac{R_1}{R_k}, \quad k = 2, 3, \cdots, K, \tag{31}$$

and if the globally uniform distribution pattern is adopted, i.e., $M_1/R_1 \approx M_2/R_2 \approx \cdots \approx M_k/R_k$, then it holds

$$\gamma_k \approx \gamma_1 \left(\frac{R_1}{R_k}\right)^{1-\frac{1}{d}}, \quad k = 2, 3, \cdots, K. \tag{32}$$

4.4. Adaptively determining the loss weighting parameters through normalization

The choice of the weighting parameters in the loss function of neural network methods may significantly affect the performance of neural network methods. Inspired by the work of [41], we propose to determine the loss weighting parameters based on the magnitudes of elements of the augmented matrix combining the least squares coefficient matrix with the target vector for the proposed Multi-TransNet method. To be specific, let us again take the case of $K = 2$ for explanation, and the weighting parameters in the loss function (19) will be calculated through the following *normalization*:

$$\begin{aligned}
\lambda_k &= \frac{1}{\max\limits_{\substack{1\leqslant i\leqslant N_k \\ 1\leqslant j\leqslant M_k+2}} \left|\left(\mathcal{L}\left(\boldsymbol{\phi}_k^{\top}\left(\boldsymbol{X}_k\right)\right) \,\vdots\, f\left(\boldsymbol{X}_k\right)\right)_{i,j}\right|}, \quad k = 1, 2, \\
\lambda_g &= \frac{1}{\max\limits_{\substack{1\leqslant i\leqslant N_g \\ 1\leqslant j\leqslant M_2+2}} \left|\left(\mathcal{B}\left(\boldsymbol{\phi}_2^{\top}\left(\boldsymbol{X}_g\right)\right) \,\vdots\, g\left(\boldsymbol{X}_g\right)\right)_{i,j}\right|}, \\
\lambda_{h,1} &= \frac{1}{\max\limits_{\substack{1\leqslant i\leqslant N_\Gamma \\ 1\leqslant j\leqslant M+3}} \left|\left(\boldsymbol{\phi}_1^{\top}\left(\boldsymbol{X}_\Gamma\right) \,\vdots\, -\boldsymbol{\phi}_2^{\top}\left(\boldsymbol{X}_\Gamma\right) \,\vdots\, h_1\left(\boldsymbol{X}_\Gamma\right)\right)_{i,j}\right|}, \\
\lambda_{h,2} &= \frac{1}{\max\limits_{\substack{1\leqslant i\leqslant N_\Gamma \\ 1\leqslant j\leqslant M+3}} \left|\left(\mathcal{J}\left(\boldsymbol{\phi}_1^{\top}\left(\boldsymbol{X}_\Gamma\right)\right)\cdot\boldsymbol{n} \,\vdots\, -\mathcal{J}\left(\boldsymbol{\phi}_2^{\top}\left(\boldsymbol{X}_\Gamma\right)\right)\cdot\boldsymbol{n} \,\vdots\, h_2\left(\boldsymbol{X}_\Gamma\right)\right)_{i,j}\right|},
\end{aligned} \tag{33}$$

where the dashed vertical lines denote the matrix augmentation operation. The extension of such normalization approach to the Multi-TransNet with multiple interfaces and K subdomains is again straightforward. We also note the approach developed in [41] applies the normalization to the least squares coefficient matrix only for obtaining the loss weighting parameters, i.e., removing the parts involving f, g, h_1 and h_2 in the case of (33).

Finally, we summarize in Algorithm 2 the main steps of the proposed Multi-TransNet method for solving the elliptic interface problem (1), i.e., compute its Multi-TransNet solution (22).

Algorithm 2: The Multi-TransNet method for solving the elliptic interface problem (1).

Input: The total number of hidden-layer neurons M.
Output: The output-layer parameters $\{\boldsymbol{\alpha}_k\}_{k=1}^K$ of Multi-TransNet

1 Select appropriately the parameters $\{(\boldsymbol{x}_c^{(k)}, R_k)\}_{k=1}^K$ for all subdomain TransNets according to the location and radius magnitude of the K subdomains $\{\Omega_k\}_k^K$.
2 Calculate the number of hidden-layer neurons for all subdomain TransNets, $\{M_k\}_{k=1}^K$, by using (25) with $\sum_{k=1}^K M_k = M$.
3 Generate the location parameters of hidden-layer neurons for all subdomain TransNets, $\{(\boldsymbol{a}_m^{(1)}, r_m^{(1)})\}_{m=1}^{M_1}$, …, $\{(\boldsymbol{a}_m^{(K)}, r_m^{(K)})\}_{m=1}^{M_K}$ according to the sampling process described in Theorem 2.
4 Determine the shape parameters of hidden-layer neurons for all subdomain TransNets, $\{\gamma_k\}_{k=1}^K$, by using the empirical formula-based prediction strategy.
5 Construct the neural basis functions for all subdomain TransNets, $\{\boldsymbol{\phi}_k\}_{k=1}^K$.
6 Sample the training/collocation points (randomly or uniformly) on subdomains, domain boundaries and interfaces.
7 Assemble the feature matrix and target vector (as done in (19)) and adaptively determine the weighting parameters for all terms in the loss function with the normalization technique (as done in (16) and (33)).
8 Find the output-layer parameters $\{\boldsymbol{\alpha}_k\}_{k=1}^K$ by minimizing the loss function (18) (i.e., solving a linear least squares problem).

5. Numerical experiments

To demonstrate the superior accuracy, efficiency and robustness of the proposed Multi-TransNet method, we perform abundant numerical tests in this section, including ablation studies and comparison with recent neural network methods and traditional numerical techniques for solving elliptic interface problems.

The hyperbolic tangent *tanh* is adopted as the activation as in [42] owing to its good smoothness. The training/collocation points are sampled uniformly in the computational domain (including interior domains, boundaries and/or interfaces), and the test points are generated via the Latin hypercube sampling method [43], which ensures that the test points are more evenly distributed across the range of each dimension. The number of training/collocation points is determined by the spacing size due to uniform sampling adopted (though the proposed Multi-TransNet method is mesh-free), and the number of test points is set to 2^d times as large as that of training/collocation points, where d is the problem dimension. When the golden-section search algorithm (Algorithm 1) is used, we set the initial search interval to be $[0, 5]$ and the number of search iteration to be 7. For measurement of the accuracy of a numerical solution $\boldsymbol{u}$, we use the discrete relative L_2 error (referred to as RL_2 for short) and the discrete relative L_∞ error (referred to as RL_∞ for short) over all the test points, which are defined as

$$\mathrm{RL}_2(\boldsymbol{u}) = \frac{\|\boldsymbol{u} - \boldsymbol{u}_{\text{exact}}\|_2}{\|\boldsymbol{u}_{\text{exact}}\|_2}, \quad \mathrm{RL}_\infty(\boldsymbol{u}) = \frac{\|\boldsymbol{u} - \boldsymbol{u}_{\text{exact}}\|_\infty}{\|\boldsymbol{u}_{\text{exact}}\|_\infty}.$$

The API *torch.linalg.lstsq* from PyTorch is called for solving the least squares problems. All the experiments are implemented on an Ubuntu 20.04.6 LTS server with a 3.00-GHz Intel Xeon Gold 6248R CPU and a NVIDIA GeForce RTX 4090 GPU. All experimental results are obtained by repeating 10 runs, removing two extrema and then taking the averages of the remaining to diminish the influence of randomness. The right-hand terms of PDEs, boundary conditions and/or interface conditions in each example can be derived from the given exact solution, and hence are not listed separately.

5.1. Ablation studies

In this subsection, we perform systematic ablation studies to verify effectiveness and benefits of some important components of the TransNet and the proposed Multi-TransNet methods.

5.1.1. TransNet

In this subsection, we test the TransNet method by solving the classic 2D Poisson problem defined in the domain $\Omega = [0, 2]^2$:

$$\begin{cases} \Delta u = f, & (x, y) \in \Omega, \\ u = g, & (x, y) \in \partial\Omega. \end{cases} \tag{34}$$

The exact solution is taken as $u(x, y) = \sin x \sin y$. We uniformly sample 49^2 points in the interior and 51×4 points on the boundary, with an overall spacing of approximately 0.04, for training.

Benefits of translating and scaling hidden-layer neurons We numerically investigate the effects of translating and scaling hidden-layer neurons of TransNet discussed in Subsection 2.2. For this propose, we design four combinations for $B_R(\boldsymbol{x}_c)$ according to whether to translate to the center of the domain Ω and whether to scale to cover the domain: (I) Not translating-not covering domain, (II) Not translating-covering domain, (III) Translating-not covering domain, and (IV) Translating-covering domain. The specific settings for parameters $\boldsymbol{x}_c$ and R are listed in Table 1, and their illustration is shown in Fig. 6. For all the four settings, the number of hidden-layer neurons of the TransNet M is gradually increased from 200 to 1000 by an increment of 200 each time, and the corresponding shape parameter γ is obtained by using the training loss-based optimization strategy for TransNet presented in Section 3. All results of the

Table 1
Parameter settings corresponding to different combinations of translating and scaling for the problem (34).

Translating & scaling	$\boldsymbol{x}_c$	R
(I) Not translating-not covering domain	(0, 0)	2.5
(II) Not translating-covering domain	(0, 0)	3.0
(III) Translating-not covering domain	(1, 1)	1.0
(IV) Translating-covering domain	**(1, 1)**	**1.5**

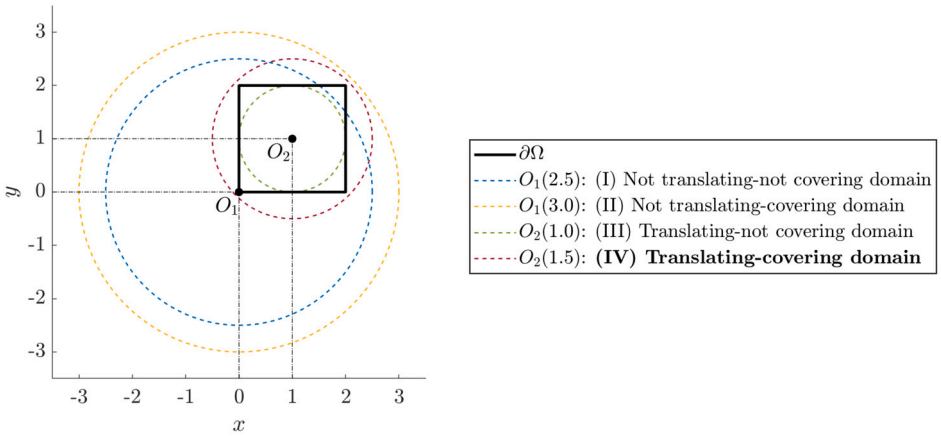

Fig. 6. Illustration of different parameter settings for the problem (34) in Table 1.

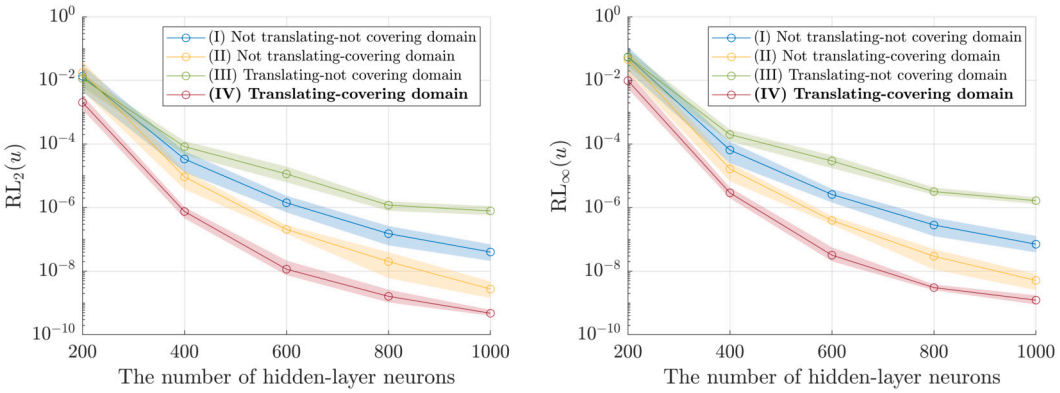

Fig. 7. Comparisons of the relative L_2 (left) and L_∞ (right) errors of numerical solutions produced by the TransNets with different combinations of translating and scaling parameters (see Table 1) for the problem (34).

relative numerical errors are shown in Fig. 7. First, we observe that both the relative L_2 and L_∞ errors of all four combinations rapidly and steadily decay as the number of hidden-layer neurons is growing. The setting of (IV) Translating-covering domain performs the best, in which the errors drop from the level of $O(10^{-2})$ to the level of $O(10^{-9})$ when M increases from 200 to 1000. Subsequently, with translating or not translating, the accuracy of the case of covering domain is remarkably better than that of the case of not covering domain, which is more and more noticeable along the growing of the number of hidden-layer neurons. These phenomena manifest that scaling to ensure that the ball $B_R(\boldsymbol{x}_c)$ covers the domain is rather crucial to the performance of TransNet. Furthermore, with covering domain, the case of translating markedly outperforms the case of not translating. These results clearly demonstrate benefits of translating and scaling hidden-layer neurons for TransNet.

Effectiveness of the empirical formula-based prediction strategy We compare the solution accuracy of the TransNets with the shape parameter γ determined by the training loss-based optimization strategy and by the empirical formula-based prediction strategy (see Section 3), respectively. The ball $B_R(\boldsymbol{x}_c)$ centered at $\boldsymbol{x}_c = (1, 1)$ with radius $R = 1.5$ (the setting of (IV) Translating-covering domain in Table 1) is used to cover the domain Ω and generate the hidden-layer neurons of the TransNet. The number of hidden-layer neurons M is again gradually increased from 200 to 1000 by an increment of 200 each time. For the training loss-based optimization strategy, the optimal shape parameter with respect to each number of hidden-layer neurons is obtained by the golden-section search algorithm. For the empirical formula-based prediction strategy, the empirical constant C in the empirical formula (13) is first estimated using the optimization strategy for the TransNet with 200 hidden-layer neurons in the preprocessing step (it is found $C \approx 8.4779\text{e-}2$), and

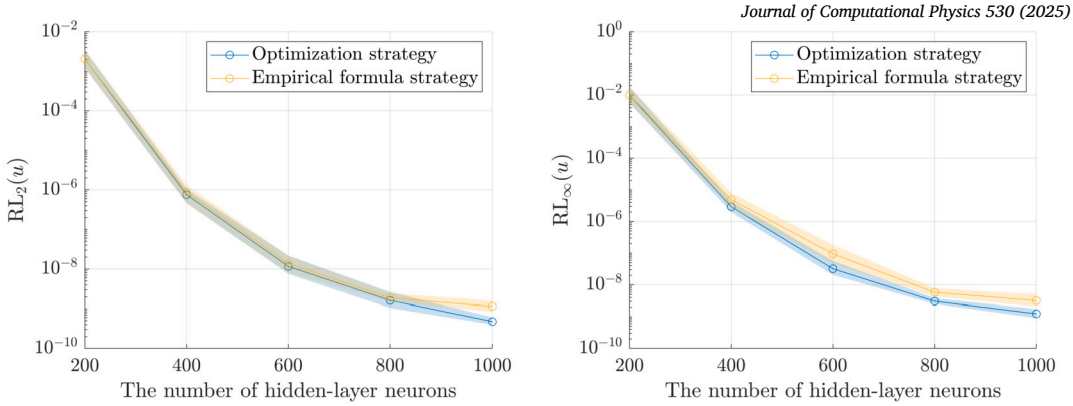

Fig. 8. Comparisons of the relative L_2 (left) and L_∞ (right) errors of numerical solutions produced by the TransNets with the shape parameter γ determined by the optimization strategy and by the empirical formula strategy for the problem (34).

then the shape parameters of the TransNets with other numbers of hidden-layer neurons are automatically calculated in a completely explicit manner. The comparison results on relative errors are shown in Fig. 8. It is seen that the accuracy of the TransNets produced by the empirical formula strategy is similar to that of the TransNets by the optimization strategy and marginally lower when the number of hidden-layer neurons reaches 1000. Hence, the proposed empirical formula-based prediction strategy is as effective as the training loss-based optimization strategy for TransNet, in addition to its high efficiency.

5.1.2. Multi-TransNet

In this subsection, we test the Multi-TransNet method by solving the following diffusion interface problem defined in the domain $\Omega = [0,2]^2$:

$$\begin{cases} -\nabla\cdot(\beta\nabla u) = f, & (x,y)\in\Omega_1\cup\Omega_2, \\ [u] = h_1, & (x,y)\in\Gamma, \\ [\beta\nabla u\cdot\boldsymbol{n}] = h_2, & (x,y)\in\Gamma, \\ u = g, & (x,y)\in\partial\Omega, \end{cases} \tag{35}$$

where the interface is a circle $\Gamma = \{(x,y)\mid (x-1)^2+(y-1)^2 = 0.5\}$. It divides Ω into two subdomains (i.e., $K=2$) Ω_1 (inside) and Ω_2 (outside). The exact solution is taken as $u(x,y) = \chi_{\Omega_1}\sin x\sin y + \chi_{\Omega_2}\cos x\cos y$, and the diffusion coefficient is defined by the piecewise constant $\beta(x,y) = 1\chi_{\Omega_1} + 10\chi_{\Omega_2}$. The proposed Multi-TransNet method clearly will consist of two subdomain TransNets u_1^{NN} (associated to Ω_1) and u_2^{NN} (associated to Ω_2). Inspired by the ablation studies for single TransNet in the previous subsection, we also apply translating and scaling to the two subdomain TransNets so that the two balls $B_{R_1}(\boldsymbol{x}_c^{(1)})$ and $B_{R_2}(\boldsymbol{x}_c^{(2)})$ can properly cover Ω_1 and Ω_2 respectively. To be detailed, the following parameters are used: $\boldsymbol{x}_c^{(1)} = (1,1)$, $R_1 = 1$ and $\boldsymbol{x}_c^{(2)} = (1,1)$, $R_2 = 1.5$. We uniformly sample 49^2 points in the interior, 51×4 points on the boundary, and 120 points on the interface, with an overall spacing of approximately 0.04, for training.

Effect of globally uniform neuron distribution We test and compare the performance of the Multi-TransNet method under the approach of equal assignment of the number of hidden-layer neurons among the subdomains (i.e., $M_1 = M_2$) and the approach of globally uniform neuron distribution across the entire computational domain (i.e., $M_1/R_1 = M_2/R_2$). The total number of hidden-layer neurons $M = M_1 + M_2$ of the Multi-TransNet is gradually increased from 400 to 1200 by an increment of 200 each time, and corresponding shape parameters (γ_1,γ_2) of the subdomain TransNets are obtained by using the training loss-based optimization strategy for Multi-TransNet in Subsection 4.3. The performance comparisons in terms of the relative errors are illustrated in Fig. 9. It is clearly observed that the globally uniform neuron distribution approach significantly prevails over the equal assignment approach (the errors of the former are approximately 10 times smaller than those of the latter). Although this is just one specific example, it implies that the globally uniform neuron distribution approach for Multi-TransNet is able to bring better transferability and accuracy in general.

Effectiveness of the empirical formula-based prediction strategy We compare the solution accuracy of the Multi-TransNet with the shape parameters (γ_1,γ_2) determined by the training loss-based optimization strategy and the empirical formula-based prediction strategy (see Subsection 4.3), respectively. The total number of hidden-layer neurons of the Multi-TransNet M is again gradually increased from 400 to 1200 by an increment of 200 each time, and the globally uniform neuron distribution approach is used for assignment of the numbers of hidden-layer neurons among the two subdomain TransNets. For the training loss-based optimization strategy, the optimal shape parameters with respect to each total number of hidden-layer neurons are obtained by the golden-section search algorithm. For the empirical formula-based prediction strategy, an estimated value 7.3037e-2 for the empirical constant C in the

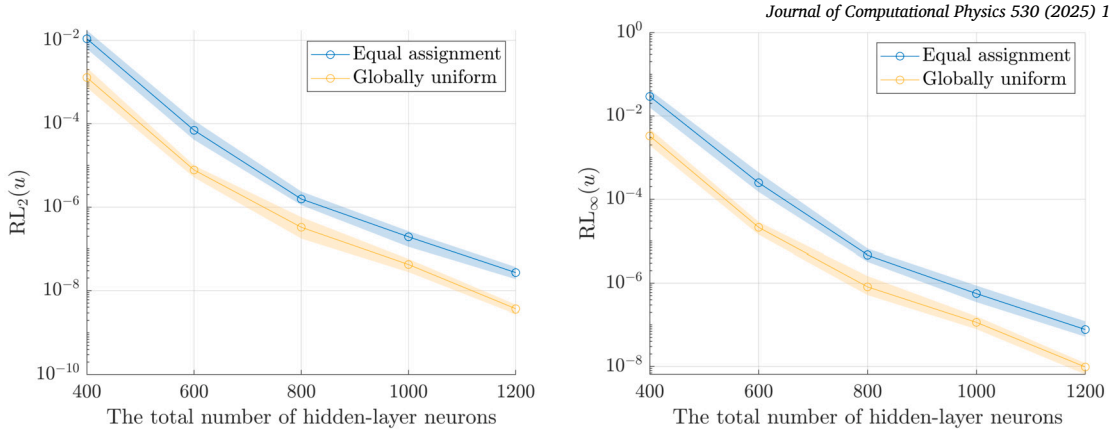

Fig. 9. Comparisons of the relative L_2 (left) and L_∞ (right) numerical errors of solutions produced by the Multi-TransNet with the approach of equal assignment of the number of hidden-layer neurons among the subdomains and with the approach of globally uniform neuron distribution across the entire computational domain for the problem (35).

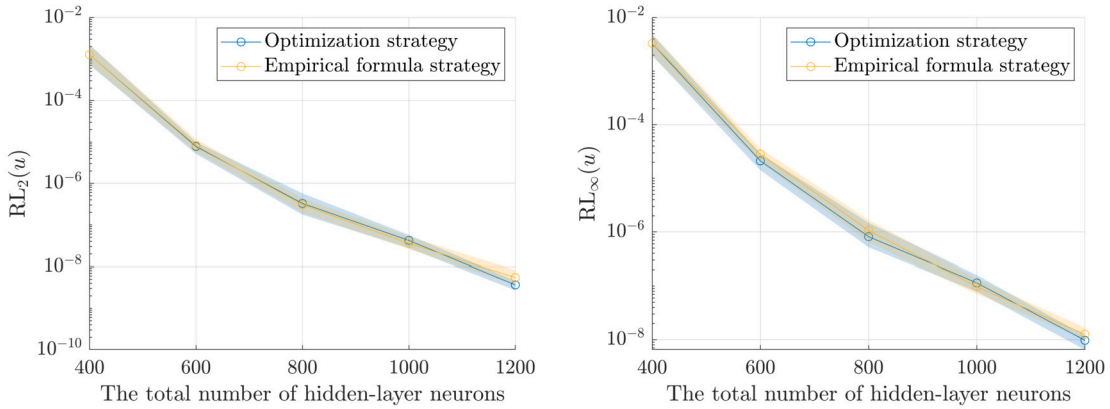

Fig. 10. Comparisons of the relative L_2 (left) and L_∞ (right) errors of numerical solutions produced by the Multi-TransNet with the shape parameters (γ_1, γ_2) determined by the optimization strategy and by the empirical formula strategy for the problem (35).

extended empirical formula (26) is first found by applying the optimization strategy to the Multi-TransNet with totally 400 hidden-layer neurons in the preprocessing step, and then the shape parameters of the Multi-TransNet with other total numbers of hidden-layer neurons are automatically calculated. Their comparison results on the relative errors are shown in Fig. 10. It is observed that the accuracies produced by both strategies are nearly the same, which demonstrate that the proposed empirical formula-based prediction strategy for shape parameters is effective and efficient for the Multi-TransNet method.

Impact of the weighting parameters in the loss function Regarding the impact of the weighting parameters of the loss function (i.e., λ_1, λ_2, λ_g, $\lambda_{h,1}$, $\lambda_{h,2}$ in (16)) to the Multi-TransNet solution, we compare three choices of determining the loss weighting parameters: (I) Setting each to one (i.e., equal weighting balances); (II) Normalizing the least squares coefficient matrix only, and (III) Normalizing the least squares augmented matrix, i.e., (33). Note that except the difference in the loss weighting parameters, all other parameter settings of Multi-TransNet are the same. For example, the globally uniform neuron distribution strategy is used for assignment of subdomain hidden-layer neurons and the shape parameters (γ_1, γ_2) are determined by the training loss-based optimization strategy under the loss weighting parameters choice (I). The comparison results are shown in Fig. 11. It is clearly observed that the relative errors produced under the choice (III) are always the best, especially when the total number of hidden-layer neurons is relatively small. The performance of the choice (II) is slightly better than that of the choice (I).

5.2. Applications of the Multi-TransNet to typical elliptic interface problems

In this subsection, we apply the proposed Multi-TransNet to a series of typical elliptic interface problems in two and three dimensions to demonstrate its outstanding performance in terms of superior accuracy, efficiency and robustness, including a 2D Stokes interface problem with a circular interface, a 2D diffusion interface problem with multiple interfaces, a 3D elasticity interface problem with an ellipsoidal interface, and a 3D diffusion interface problem with a convoluted immersed interface. The default configuration of the following Multi-TransNet is a combination of properly translating and scaling hidden-layer neurons based on subdomains, the

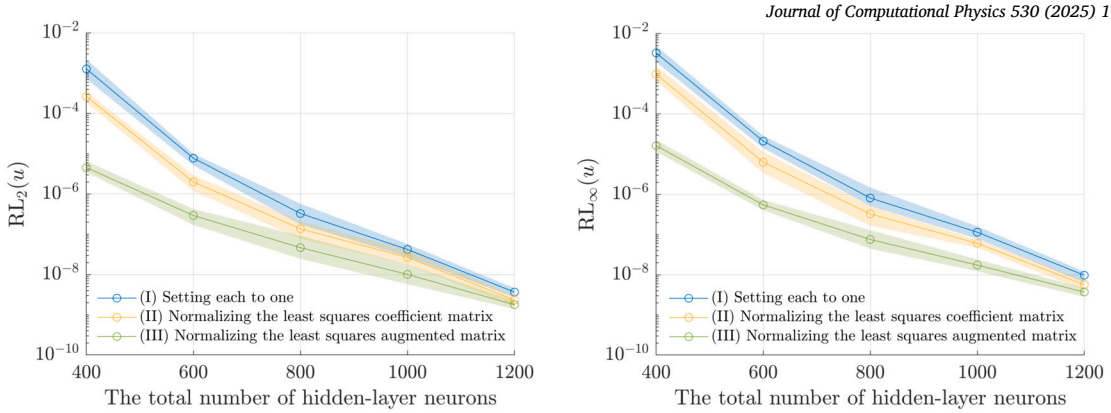

Fig. 11. Comparisons of the relative L_2 (left) and L_∞ (right) errors of numerical solutions produced by the Multi-TransNet under three different choices of the loss weighting parameters for the problem (35).

globally uniform neuron distribution, the empirical formula-based prediction strategy for the shape parameters, and the normalization approach for weighting parameters in the loss function.

5.2.1. A 2D Stokes interface problem with a circular interface

In this example, we consider the two-phase Stokes interface problem [44] in fluid mechanics field. The domain $\Omega = [-2,2]^2$ is divided into two subdomains Ω_1 (the inside one) and Ω_2 (the outside one) by a circular interface $\Gamma = \{(x,y) \mid x^2 + y^2 = 1\}$. The problem is described as follows:

$$\begin{cases} -\mu\Delta \boldsymbol{u} + \nabla p = \boldsymbol{f}, & (x,y) \in \Omega_1 \cup \Omega_2, \\ \nabla \cdot \boldsymbol{u} = 0, & (x,y) \in \Omega_1 \cup \Omega_2, \\ [\boldsymbol{u}] = \boldsymbol{0}, & (x,y) \in \Gamma, \\ [\boldsymbol{\sigma}(\boldsymbol{u},p)\boldsymbol{n}] = \boldsymbol{h}_2, & (x,y) \in \Gamma, \\ \boldsymbol{u} = \boldsymbol{g}, & (x,y) \in \partial\Omega, \end{cases} \tag{36}$$

where the viscosity $\mu(x,y) = \mu_k$ for $(x,y) \in \Omega_k$ $(k = 1,2)$, the velocity vector $\boldsymbol{u} = (u,v)^\top$ and the stress tensor $\boldsymbol{\sigma}(\boldsymbol{u},p) = -p\boldsymbol{I} + \mu\left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^\top\right)$. Its exact solution is given by

$$\begin{aligned} u(x,y) &= \begin{cases} \frac{y}{4}\left(x^2+y^2\right), & (x,y) \in \Omega_1, \\ \frac{y}{\sqrt{x^2+y^2}} - \frac{3y}{4}, & (x,y) \in \Omega_2, \end{cases} \\ v(x,y) &= \begin{cases} -\frac{xy^2}{4}, & (x,y) \in \Omega_1, \\ -\frac{x}{\sqrt{x^2+y^2}} + \frac{x}{4}\left(3+x^2\right), & (x,y) \in \Omega_2, \end{cases} \\ p(x,y) &= \begin{cases} 5.0, & (x,y) \in \Omega_1, \\ \left(-\frac{3}{4}x^3 + \frac{3}{8}x\right)y, & (x,y) \in \Omega_2. \end{cases} \end{aligned} \tag{37}$$

Note that u, v are continuous but nonsmooth across the interface while p is discontinuous as shown in the top row of Fig. 12. The viscosity $\mu_1 = 1$ is fixed and different values will be chosen for μ_2 to reflect various ratio contrast (i.e., μ_1/μ_2) cases. We sample $N_f = 149^2$ (in the interior), $N_g = 301 \times 4$ (on the boundary) and $N_\Gamma = 1000$ (on the interface) training points, and use two subdomain TransNets with a combination of translating and scaling parameters $(\boldsymbol{x}_c^{(1)}, R_1) = ((0.0,0.0), 1.25)$, $(\boldsymbol{x}_c^{(2)}, R_2) = ((0.0,0.0), 3.0)$ for the Multi-TransNet method to approximate the solution of the problem (36).

The bottom row of Fig. 12 illustrates the pointwise absolute errors of the numerical solution produced by the Multi-TransNet method with totally $M = M_1 + M_2 = 1000$ hidden-layer neurons under the training loss-based optimization strategy for optimal shape parameters (γ_1, γ_2) in the low-contrast case of the viscosity $\mu_2 = 10$. Meanwhile, an estimated value 8.6107e-2 is obtained for the empirical constant C in the extended empirical formula (26) for Multi-TransNet. Subsequently, the empirical formula-based prediction strategy is employed to determine appropriate shape parameters for the Multi-TransNet with other numbers of hidden-layer neurons, which will also be used to solve other viscosity contrast cases. The numerical results for $\mu_1/\mu_2 = 10^{-1}, 10^{-2}, 10^{-3}$ and 10^{-4} are illustrated in Fig. 13, from which it is clearly observed that the relative L_2 and L_∞ errors of the Multi-TransNet solutions quickly decline when the total number of hidden-layer neurons gradually increases from $M = 1000, 2000, 4000$ to $M = 6000$, and finally reach a level of $O(10^{-10})$ for the velocity components u and v. Furthermore, the relative errors of both velocity components remain flat regardless of the viscosity contrast, while that of the pressure decreases with the lowering of the viscosity contrast (note

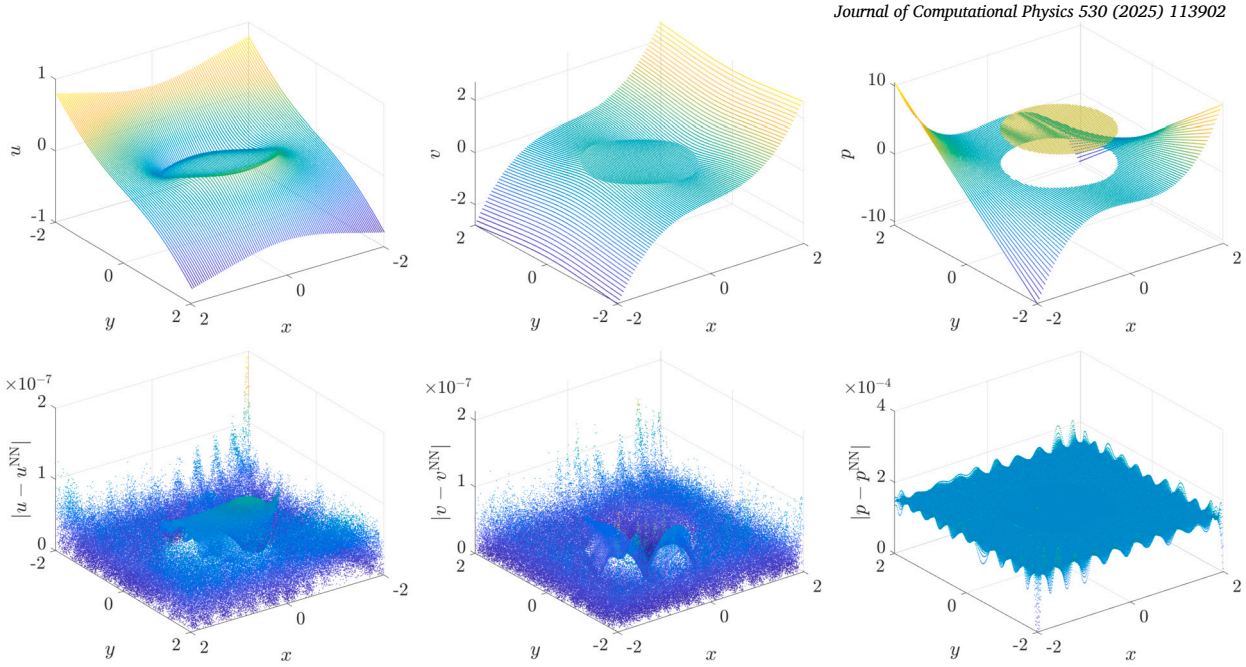

Fig. 12. Top row: the exact solution of the 2D Stokes interface problem (36); Bottom row: the pointwise absolute errors of the numerical solution, when the viscosity $(\mu_1, \mu_2) = (1, 10)$, produced by the Multi-TransNet with totally 1000 hidden-layer neurons. From left to right: u, v and p.

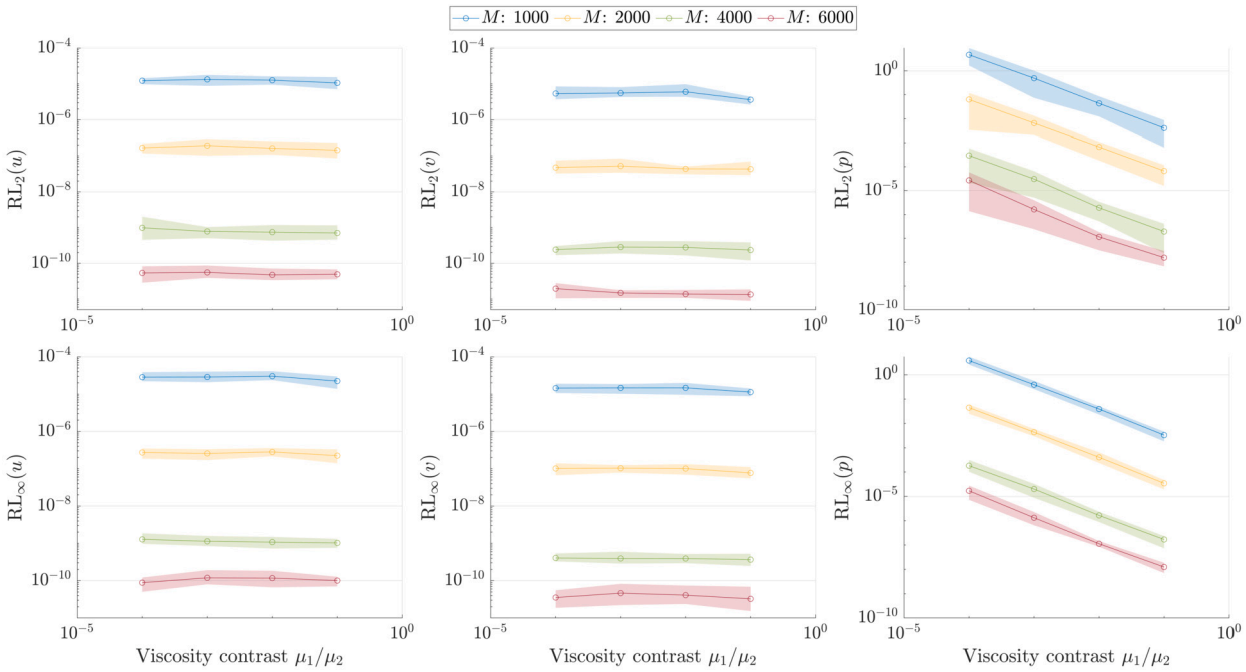

Fig. 13. The relative L_2 (top row) and L_∞ (bottom row) errors of the numerical solutions produced by the Multi-TransNet method with different total numbers of hidden-layer neurons ($M = 1000, 2000, 4000, 6000$) for the 2D Stokes interface problem (36) with different viscosity contrasts ($\mu_1/\mu_2 = 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$). From left to right: u, v and p.

similar behaviors were also observed on the pressure solutions by the traditional numerical method [44] and the Random Feature Method (RFM) [41] for this example). Fig. 14 reports the average running times of the Multi-TransNet method with respect to different total numbers of hidden-layer neurons. We observe that the system assembling time seems linearly proportional to the total number of hidden-layer neurons, but the least squares solving time increases much faster along with the growing of the total number of hidden-layer neurons.

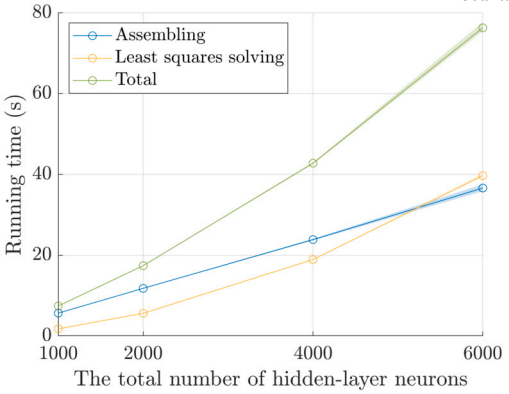

Fig. 14. The average running times (measured in seconds) of the Multi-TransNet method with different total numbers of hidden-layer neurons ($M = 1000, 2000, 4000, 6000$) for the 2D Stokes interface problem (36).

Table 2
Comparison of the performance of the Multi-TransNet and the RFM [41] for the 2D Stokes interface problem (36) with different viscosity contrasts ($\mu_1/\mu_2 = 10^{-1}, 10^{-2}, 10^{-3}$). Note that #DOFs $= 3M$ for the Multi-TransNet method in this problem.

μ_1/μ_2	Method	#Constraints	#DOFs	$RL_2(u)$	$RL_2(v)$	$RL_2(p)$
10^{-1}	RFM	84,801	9,600	6.92e-6	1.71e-6	3.33e-4
			38,400	1.18e-8	3.30e-9	4.84e-8
	Multi-TransNet	73,012	6,000	1.39e-7	4.04e-8	6.45e-5
			18,000	4.88e-11	1.33e-11	1.43e-8
10^{-2}	RFM	84,801	9,600	4.93e-6	1.48e-6	2.63e-3
			38,400	1.52e-8	3.21e-9	5.99e-7
	Multi-TransNet	73,012	6,000	1.58e-7	4.52e-8	6.89e-4
			18,000	4.59e-11	1.37e-11	1.20e-7
10^{-3}	RFM	84,801	9,600	4.49e-6	1.44e-6	3.31e-2
			38,400	5.97e-9	1.64e-9	2.76e-6
	Multi-TransNet	73,012	6,000	1.90e-7	5.02e-8	6.16e-3
			18,000	5.12e-11	1.51e-11	1.50e-6

Finally, we compare the performance of the Multi-TransNet with the RFM in terms of accuracy and efficiency, and the results are listed in Table 2. We observe that both neural network methods exhibit good robustness for different viscosity contrasts. Nonetheless, it is easy to see that the Multi-TransNet strikingly outperforms the RFM regardless of the number of constraints (i.e., the row number of the resulting least squares system) and the number of DOFs (i.e., the column number of the resulting least squares system, or the product of the total number of hidden-layer neurons and the number of unknown variables). In particular, the relative L_2 errors of the Multi-TransNet with even smaller number of DOFs are several magnitudes smaller than those of the RFM for the velocity components u and v.

5.2.2. A 2D diffusion interface problem with multiple interfaces

In this example, we consider the two-dimensional diffusion interface problem with multiple interfaces [45]. The domain $\Omega = [-1, 1]^2$ is divided into four subdomains Ω_1, Ω_2, Ω_3 and Ω_4 (from the inside to the outside) by the following three closed curves:

$$\Gamma_1 = \left\{(x, y) \mid x^2 + y^2 = 0.2^2\right\},$$
$$\Gamma_2 = \left\{(x, y) \mid x^2 + y^2 = (0.5 - 0.1\cos(5\theta))^2\right\},$$
$$\Gamma_3 = \left\{(x, y) \mid x^2 + y^2 = 0.8^2\right\},$$

as plotted in the left of Fig. 15, where $\theta \in [0, 2\pi)$ is the angle between the positive x-axis and the segment connecting the point (x, y) and the origin. The problem is formulated as

$$\begin{cases} -\nabla \cdot (\beta \nabla u) = f, & (x, y) \in \Omega_1 \cup \Omega_2 \cup \Omega_3 \cup \Omega_4, \\ [u] = h_1, & (x, y) \in \Gamma_i, \quad i = 1, 2, 3, \\ [\beta \nabla u \cdot \boldsymbol{n}_i] = h_2, & (x, y) \in \Gamma_i, \quad i = 1, 2, 3, \\ u = g, & (x, y) \in \partial\Omega, \end{cases} \tag{38}$$

where $\beta(x, y) = \sum_{k=1}^{4} \beta_k \chi_{\Omega_k}$. The corresponding exact solution is given by

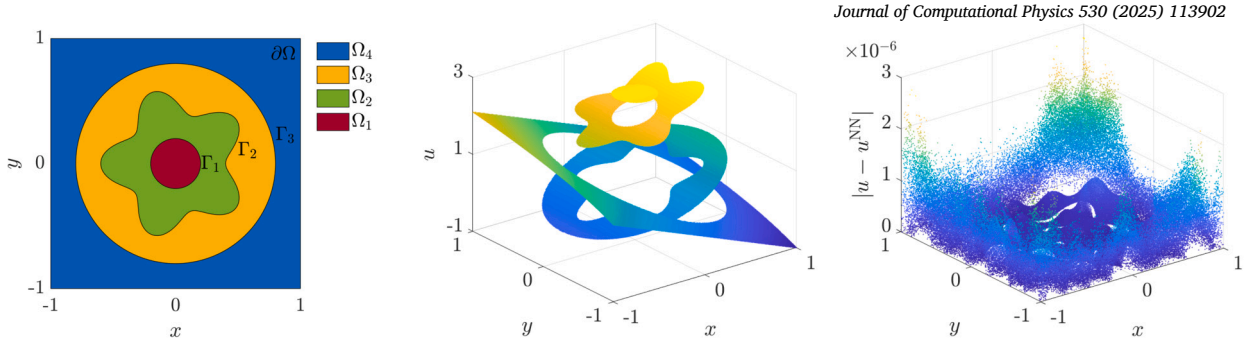

Fig. 15. Left: illustration of the partition of the domain into four subdomains by three interfaces; Middle: the exact solution of the 2D diffusion interface problem (38); Right: the pointwise absolute errors of numerical solution, when the diffusion coefficient $(\beta_1, \beta_2, \beta_3, \beta_4) = (1, 10, 10^2, 10^3)$, produced by the Multi-TransNet method with totally 400 hidden-layer neurons.

Table 3
Comparison of the performance of the Multi-TransNet and the LRNN [40] for the 2D diffusion interface problem with multiple interfaces (38) under different combinations of diffusion coefficients. Note that #DOFs $= M$ for the Multi-TransNet, with the same number of constraints as those in LRNN, in this problem.

$(\beta_1, \beta_2, \beta_3, \beta_4)$	Method	#DOFs	RL_2 (u)
$(1, 10^2, 10^4, 10^6)$	LRNN	1,280	4.64e-5
	Multi-TransNet	800	2.41e-8
$(10^{-6}, 10^{-4}, 10^{-2}, 1)$	LRNN	1,280	4.44e-6
	Multi-TransNet	800	2.25e-8
$(1, 10^{-2}, 10^{-4}, 10^{-6})$	LRNN	1,280	7.68e-7
	Multi-TransNet	800	5.75e-8
$(10^6, 10^4, 10^2, 1)$	LRNN	1,280	7.16e-5
	Multi-TransNet	800	5.68e-8

$$u(x,y) = \begin{cases} \cos y + 1.8, & (x,y) \in \Omega_1, \\ e^x + 1.3, & (x,y) \in \Omega_2, \\ \sin x + 0.5, & (x,y) \in \Omega_3, \\ -x + \ln(y+2), & (x,y) \in \Omega_4, \end{cases} \tag{39}$$

as shown in the middle of Fig. 15. The diffusion coefficient $\beta_1 = 1$ is fixed and different values will be selected for $\beta_2, \beta_3, \beta_4$ to reflect various ratio contrast (i.e., β_k/β_{k+1} for $k = 1, 2, 3$) cases. We use four subdomain TransNets for the Multi-TransNet method with totally 400 hidden-layer neurons and the translating and scaling parameters $\left\{(\boldsymbol{x}_c^{(k)}, R_i)\right\}_{k=1}^4$, where $\boldsymbol{x}_c^{(k)} = (0.0, 0.0)$ for $k = 1, 2, 3, 4$, and $R_1 = 0.35$, $R_2 = 0.75$, $R_3 = 1.0$, $R_4 = 1.6$, to solve the problem (38). We uniformly sample the same number of points as those in [40], i.e., $N_f = 3000$ in the interior, $N_g = 500$ on the boundary, and $N_{\Gamma,i} = 500, i = 1, 2, 3$ on the interfaces, for training.

The case of diffusion coefficients $\beta_k = 10^{k-1}$ $(k = 2, 3, 4)$ is first taken. The right of Fig. 15 shows the pointwise absolute errors of the numerical solution produced by the Multi-TransNet method with totally $M = \sum_{k=1}^4 M_k = 400$ hidden-layer neurons under the training loss-based optimization strategy for optimal shape parameters $\{\gamma_k\}_{k=1}^4$. Meanwhile, an estimated value 4.1474e-2 is obtained for the empirical constant C in the extended empirical formula (29). Afterward, this empirical constant is employed to estimate appropriate shape parameters for the Multi-TransNet method with other numbers of hidden-layer neurons, which will also be used to solve other cases of diffusion coefficient settings $\beta_k/\beta_{k+1} = 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$ for $k = 1, 2, 3$. The corresponding numerical results are shown in Fig. 16. It is clearly observed that the relative L_2 and L_∞ errors of the Multi-TransNet solutions and gradients rapidly decay as the total number of hidden-layer neurons gradually doubles from $M = 400, 800$ to $M = 1600$, finally attaining a level of $O(10^{-9})$ for the solution u and $O(10^{-8})$ for the gradient ∇u. Moreover, all the results of the Multi-TransNet almost remain flat for the diffusion coefficients with low to high contrasts.

Finally, we compare the performance of the Multi-TransNet method with totally 800 hidden-layer neurons with the Local Randomized Neural Network (LRNN) [40] method with 1280 hidden-layer neurons for this example with different diffusion coefficient cases $(\beta_1, \beta_2, \beta_3, \beta_4)$, and the results are reported in Table 3. It is evidently noticed that the Multi-TransNet remarkably outperforms the LRNN.

5.2.3. A 3D elasticity interface problem with an ellipsoidal interface

In this example, we consider a three-dimensional elasticity interface problem with an ellipsoidal interface [46]. The domain $\Omega = [0, 1]^3$ is partitioned into two subdomains Ω_1 (inside) and Ω_2 (outside) by the following ellipsoidal surface

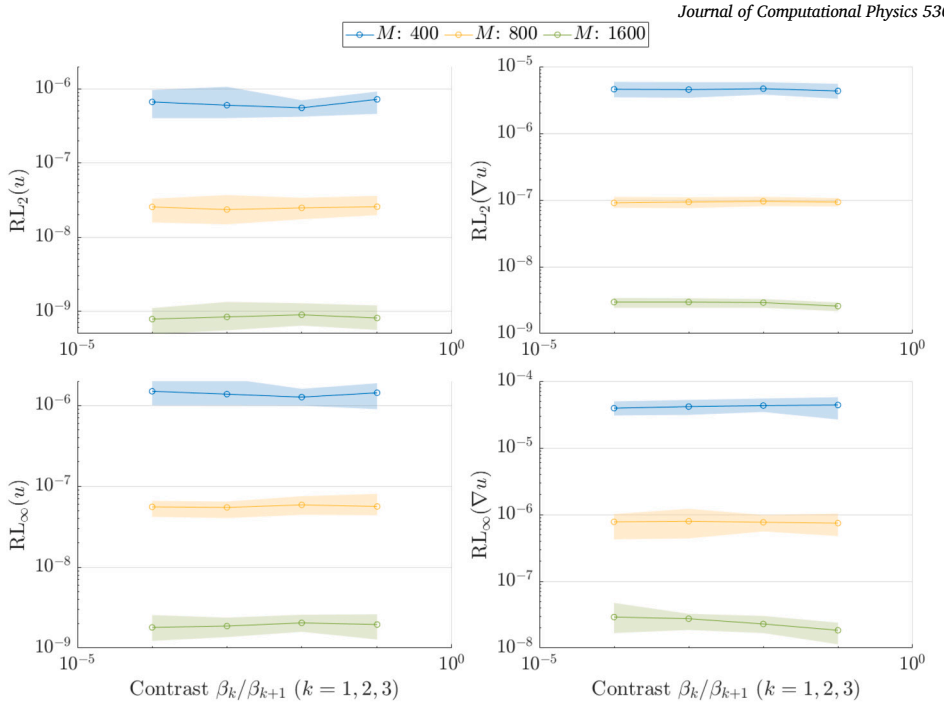

Fig. 16. The relative L_2 and L_∞ errors of the numerical solutions (left column) and their gradients (right column) produced by the Multi-TransNet with different total numbers of hidden-layer neurons ($M = 400, 800, 1600$) for the 2D diffusion interface problem (38) with the piecewise constant diffusion coefficient under different contrasts $\beta_k/\beta_{k+1} = 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$, $k = 1, 2, 3$.

$$\Gamma: \ \frac{(x-0.5)^2}{a^2} + \frac{(y-0.5)^2}{b^2} + \frac{(z-0.5)^2}{c^2} = 1,$$

where $a = 0.15, b = 0.2$ and $c = 0.25$. The problem is described by

$$\begin{cases} -\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{u}) = \boldsymbol{f}, & (x, y, z) \in \Omega_1 \cup \Omega_2, \\ [\boldsymbol{u}] = \boldsymbol{h}_1, & (x, y, z) \in \Gamma, \\ [\boldsymbol{\sigma}(\boldsymbol{u})\boldsymbol{n}] = \boldsymbol{h}_2, & (x, y, z) \in \Gamma, \\ \boldsymbol{u} = \boldsymbol{g}, & (x, y, z) \in \partial\Omega, \end{cases} \tag{40}$$

where the displacement $\boldsymbol{u} = (u, v, w)^\top$, the stress tensor $\boldsymbol{\sigma}(\boldsymbol{u}) = \lambda(\nabla \cdot \boldsymbol{u})\boldsymbol{I} + 2\mu\boldsymbol{\epsilon}(\boldsymbol{u})$, the strain tensor $\boldsymbol{\epsilon}(\boldsymbol{u}) = \left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^\top\right)/2$, the Lamé parameters $\lambda(x, y, z) = \lambda_k$ and $\mu(x, y, z) = \mu_k$ for $(x, y, z) \in \Omega_k$, $k = 1, 2$. The corresponding exact solution is chosen as

$$u(x, y, z) = \begin{cases} -\cos\left(x^2\right) \mathrm{e}^{-y^2} \sin(2\pi z), & (x, y, z) \in \Omega_1, \\ -\sin\left(x^2\right) \mathrm{e}^{y^2} \cos(2\pi z), & (x, y, z) \in \Omega_2, \end{cases}$$

$$v(x, y, z) = \begin{cases} -\cos\left(y^2\right) \mathrm{e}^{-x^2} \sin(2\pi z), & (x, y, z) \in \Omega_1, \\ -\sin\left(y^2\right) \mathrm{e}^{x^2} \cos(2\pi z), & (x, y, z) \in \Omega_2, \end{cases}$$

$$w(x, y, z) = \begin{cases} \cos\left(y^2\right) \mathrm{e}^{-z^2} \sin(2\pi x), & (x, y, z) \in \Omega_1, \\ \sin\left(y^2\right) \mathrm{e}^{z^2} \cos(2\pi x), & (x, y, z) \in \Omega_2, \end{cases}$$

which on the interface and three centered intersecting planes is plotted in the top row of Fig. 17. The Lamé parameters $\lambda_1 = 1$ and $\mu_1 = 1$ are fixed and other values will be set to reflect various contrast (i.e., λ_1/λ_2 and μ_1/μ_2) cases. Two subdomain TransNets are used for Multi-TransNet to solve the problem (40), and a combination of translating and scaling parameters is set to be $(\boldsymbol{x}_c^{(1)}, R_1) = ((0.5, 0.5, 0.5), 0.35)$ for Ω_1 and $(\boldsymbol{x}_c^{(2)}, R_2) = ((0.5, 0.5, 0.5), 1.0)$ for Ω_2. The numbers of sampling points for training are fixed to $N_f = 29^3$ (in the interior), $N_g = 31^2 \times 6$ (on the boundary) and $N_\Gamma = 60 \times 30$ (on the interface), with an overall spacing of about 0.03.

We first tackle the low-contrast case of the Lamé parameters $\lambda_2 = \mu_2 = 10$ by using the Multi-TransNet with totally 500 hidden-layer neurons and the training loss-based optimization strategy for optimal shape parameters (γ_1, γ_2). The pointwise absolute errors of the numerical solution are illustrated in the bottom row of Fig. 17. We observe that the absolute errors on the interface and intersecting planes reach the level of approximately 6e-4. An estimated value 8.1606e-2 is obtained for the empirical constant C in the extended empirical formula (26) for the Multi-TransNet. Then the empirical formula-based prediction strategy is employed

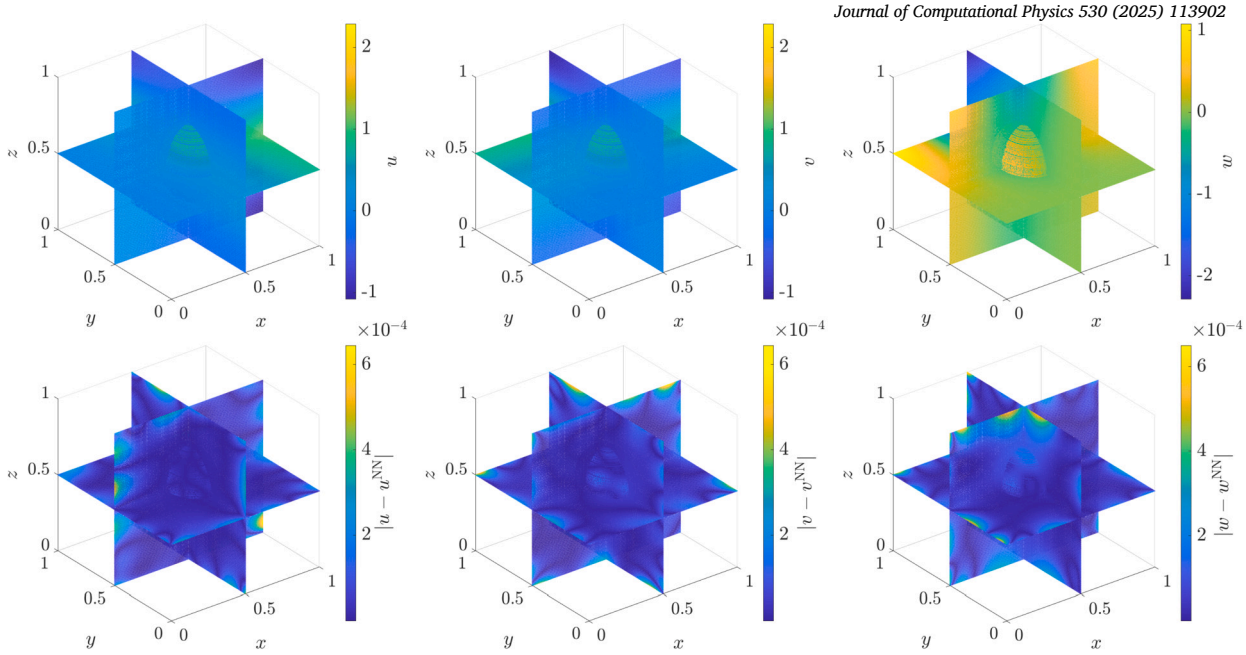

Fig. 17. Top row: the exact solution of the 3D elasticity interface problem (40) on the interface and three centered intersecting planes; Bottom row: the corresponding pointwise absolute errors of the numerical solution, when the Lamé parameters $(\lambda_1, \lambda_2) = (\mu_1, \mu_2) = (1, 10)$, produced by the Multi-TransNet method with totally 500 hidden-layer neurons. From left to right: u, v and w.

Table 4
Comparison of the performance of the Multi-TransNet and the high-order NIPFEM [46] for the 3D elasticity interface problem (40) with the high-contrast Lamé parameters $(\lambda_1, \lambda_2) = (\mu_1, \mu_2) = (1, 1000)$. Note that #DOFs $= 3M$ for the Multi-TransNet in this problem.

Method	#Constraints	#DOFs	RL_2 ($\boldsymbol{u}$)
NIPFEM	23,550	23,550	2.964e-3
	176,694	176,694	1.382e-4
	1,369,446	1,369,446	6.959e-6
Multi-TransNet	101,262	1,500	2.115e-4
		3,000	1.282e-6
		6,000	7.073e-9

to determine appropriate shape parameters for the Multi-TransNet with other numbers of hidden-layer neurons, which will be used to solve different cases of the Lamé parameter contrasts λ_1/λ_2 and μ_1/μ_2. Fig. 18 shows the numerical results for $\lambda_1/\lambda_2 = \mu_1/\mu_2$ increasing from 10^{-4} to 10^4 by a factor of 10 each time. From that, it is easy to see the relative L_2 and L_∞ errors of the Multi-TransNet solutions quickly decrease when the total number of hidden-layer neurons gradually doubles from $M = 500, 1000, 2000$ to $M = 4000$, and finally reach a level of $O(10^{-10})$ for all variables u, v and w. Furthermore, we observe that both relative errors mostly remain flat along the significant changes of the magnitude of the Lamé parameter contrasts, except that the relative L_∞ errors exhibit some fluctuations when M is taken as 4000. Fig. 19 reports the average running times of the Multi-TransNet with respect to different total numbers of hidden-layer neurons. We again find that the system assembling time doubles along with the doubling of the number of hidden-layer neurons but the least squares solving time increases faster.

Finally, we compare the performance of the Multi-TransNet method with the high-order Non-symmetric Interface Penalty Finite Element Method (NIPFEM) [46], which was recently proposed for solving the 3D elasticity interface problem with the high-contrast Lamé parameters $(\lambda_1, \lambda_2) = (\mu_1, \mu_2) = (1, 1000)$. The numerical results are reported in Table 4 and it is easy to find that Multi-TransNet easily and significantly beats the NIPFEM in terms of the number of DOFs and accuracy.

5.2.4. A 3D diffusion interface problem with a convoluted immersed interface

In this example, we consider a 3D diffusion interface problem with a convoluted interface [35] immersed in a spherical shell-shaped domain $\Omega = \{(x, y, z) \mid 0.151^2 \leqslant x^2 + y^2 + z^2 \leqslant 0.911^2\}$. The interface Γ is implicitly defined by

$$\sqrt{x^2+y^2+z^2} - r_0\left(1 + \left(\frac{x^2+y^2}{x^2+y^2+z^2}\right)^2 \sum_{k=1}^{3} a_k \cos\left(n_k\left(\arctan\left(\frac{y}{x}\right) - \theta_k\right)\right)\right) = 0, \tag{41}$$

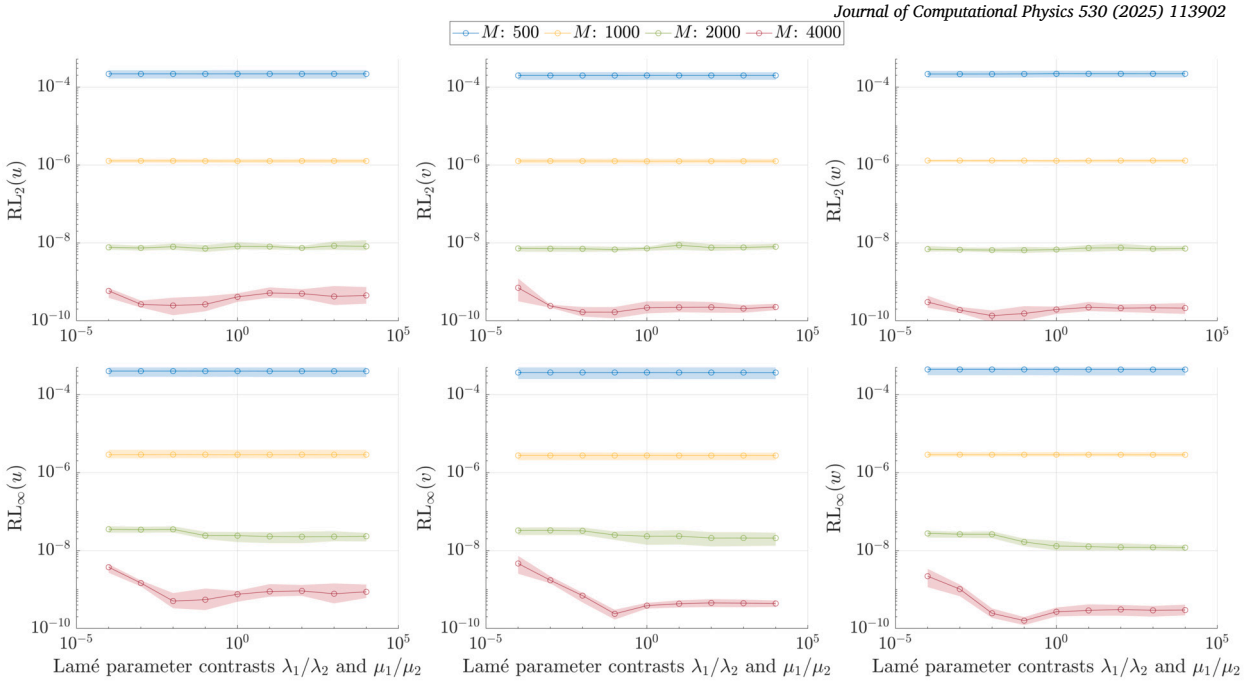

Fig. 18. The relative L_2 (top row) and L_∞ (bottom row) errors of the numerical solutions produced by the Multi-TransNet method with different total numbers of hidden-layer neurons ($M = 500, 1000, 2000, 4000$) for the 3D elasticity interface problem (40) with different Lamé parameter contrasts. From left to right: u, v and w.

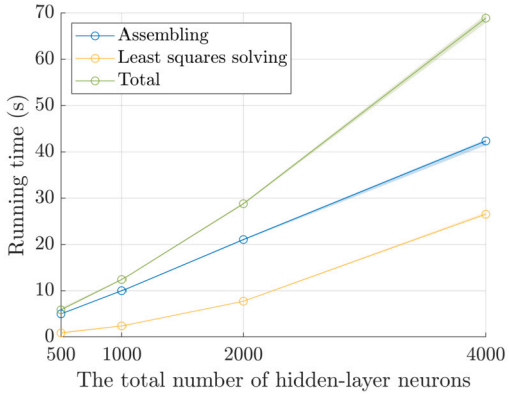

Fig. 19. The average running times (measured in seconds) of the Multi-TransNet method with different total numbers of hidden-layer neurons ($M = 500, 1000, 2000, 4000$) for the 3D elasticity interface problem (40).

with the parameters

$$r_0 = 0.483, \quad \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 0.1 \\ -0.1 \\ 0.15 \end{pmatrix}, \quad \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 7 \end{pmatrix}, \text{ and } \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 1.8 \\ 0 \end{pmatrix},$$

as plotted in Fig. 20. The problem is formulated as

$$\begin{cases} -\nabla \cdot (\beta \nabla u) = f, & (x, y, z) \in \Omega_1 \cup \Omega_2, \\ [u] = h_1, & (x, y, z) \in \Gamma, \\ [\beta \nabla u \cdot \boldsymbol{n}] = h_2, & (x, y, z) \in \Gamma, \\ u = g, & (x, y, z) \in \partial\Omega, \end{cases} \tag{42}$$

where Ω_1 denotes the interior subdomain and Ω_2 the exterior subdomain. The exact solution is chosen as

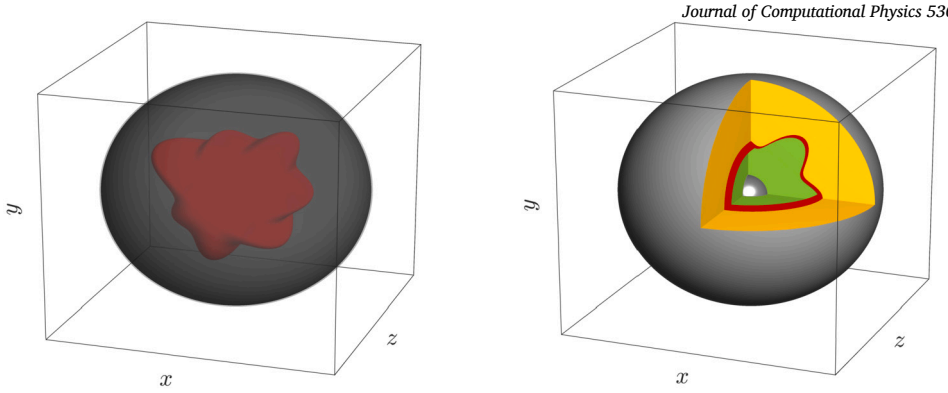

Fig. 20. Profiles of the convoluted interface (41) immersed in a spherical shell-shaped domain for the 3D diffusion interface problem (42).

Table 5
Comparison of the accuracy and running times of the Multi-TransNet method and the cusp-capturing PINN method [38] for the 3D diffusion interface problem (42) with the spatially varying diffusion coefficient (44). Note that M and #DOFs represent the number of the hidden-layer neurons and the number of trainable parameters, respectively.

Method	(M, #DOFs)	$\mathrm{RL}_2(u)$	$\mathrm{RL}_\infty(u)$	Running time (s)
cusp-capturing PINN	(25, 150)	1.10e-3	1.86e-3	55.06
	(50, 300)	2.01e-5	5.11e-5	75.12
	(100, 600)	2.10e-6	5.66e-6	91.23
Multi-TransNet	(500, 500)	3.60e-5	1.73e-4	0.33
	(1000, 1000)	1.02e-6	5.08e-6	0.76
	(2000, 2000)	1.17e-7	3.07e-7	1.70

$$u(x,y,z)=\begin{cases}\sin(2x)\cos(2y)\exp(z), & (x,y,z)\in\Omega_1,\\ \left(16\left(\frac{(y-x)}{3}\right)^5-20\left(\frac{(y-x)}{3}\right)^3+5\left(\frac{(y-x)}{3}\right)\right)\log(x+y+3)\cos(z), & (x,y,z)\in\Omega_2,\end{cases}\tag{43}$$

which on three coordinate planes (xy, xz and yz) are shown in the top row of Fig. 21.

We employ two subdomain TransNets with the respective translating and scaling parameters $(\boldsymbol{x}_c^{(1)}, R_1)=((0.0,0.0,0.0),0.75)$ and $(\boldsymbol{x}_c^{(2)}, R_2)=((0.0,0.0,0.0),1.0)$ for Multi-TransNet to solve the problem (42). The numbers of sampling points for training are set to be around $N_f=12000$ (in the interior), $N_g=2000$ (on the boundaries, 200 on the interior one and 1800 on the exterior one) and $N_\Gamma=750$ (on the interface). We first carry out the convergence test for the case of a spatially varying diffusion coefficient, and then test for the case of a piecewise constant diffusion coefficient.

Case I: with a spatially varying diffusion coefficient The diffusion coefficient is defined by

$$\beta(x,y,z)=\begin{cases}10\left(1+\frac{1}{5}\cos(2\pi(x+y))\sin(2\pi(x-y))\cos(z)\right), & (x,y,z)\in\Omega_1,\\ 1, & (x,y,z)\in\Omega_2.\end{cases}\tag{44}$$

We first use a total of 250 hidden-layer neurons for the Multi-TransNet along with the training loss-based optimization strategy for searching optimal shape parameters (γ_1,γ_2) and the pointwise absolute errors of the numerical solution on three coordinate planes are shown in the bottom row of Fig. 21. The whole golden-section searching process takes about 1.11 seconds. With the estimated value 4.4180e-2 for the empirical constant C in the empirical formula (26), the empirical formula-based prediction strategy is then employed to determine appropriate shape parameters for the Multi-TransNet with totally 500, 1000 and 2000 hidden-layer neurons to solve the problem (42). The convergence results on the relative errors of the Multi-TransNet solutions and their partial derivatives and gradients for the solution process are shown in Fig. 22. We observe that all relative L_2 and L_∞ errors rapidly and steadily decay along the doubling of the total number of hidden-layer neurons.

Furthermore, we compare the accuracy and running times of the Multi-TransNet method with the cusp-capturing PINN method [38], which was recently proposed for solving the elliptic interface problem. The numerical results are reported in Table 5, from which it is observed that the running times of the Multi-TransNet method are significantly less than those of the cusp-capturing PINN when achieving nearly the same level of relative errors. This is because the Multi-TransNet method can efficiently solve the parameters of the output layer using the well-established least-squares techniques.

Case II: with a piecewise constant diffusion coefficient The diffusion coefficient is defined to $\beta(x,y,z)=\beta_k$ for $(x,y,z)\in\Omega_k$ $(k=1,2)$. We fix $\beta_2=1$, and adopt all of the parameters of the Multi-TransNet from *Case I* to solve *Case II*, where the diffusion coefficient contrast β_1/β_2 increase from 10^{-4} to 10^4 by a factor of 10 each time. The relative L_2 and L_∞ errors of the Multi-TransNet solution

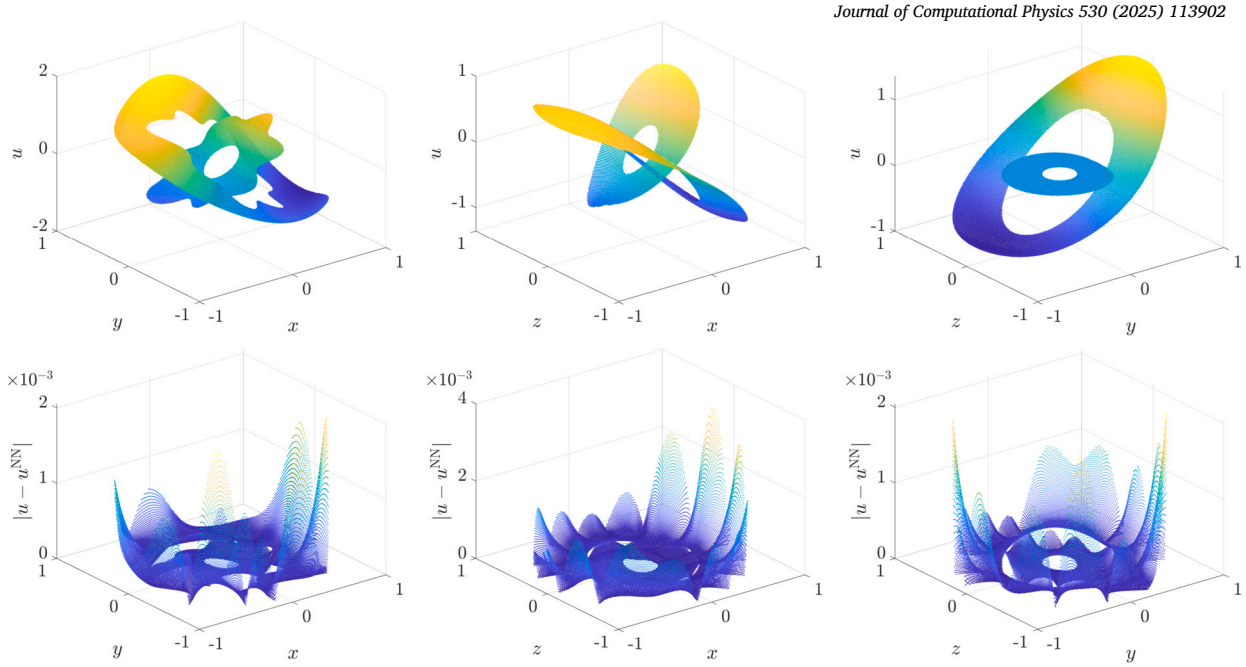

Fig. 21. Top row: the exact solution of the 3D diffusion interface problem (42) on the three coordinate planes; Bottom row: the pointwise absolute errors of the numerical solution on the three coordinate planes (in the case of the spatially varying diffusion coefficient (44)) produced by the Multi-TransNet with totally 250 hidden-layer neurons. From left to right: xy-plane, xz-plane and yz-plane.

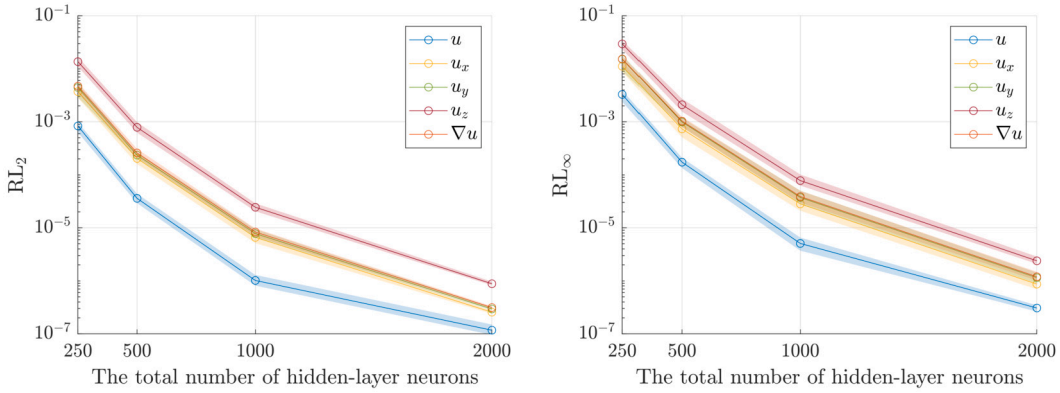

Fig. 22. The relative L_2 (left) and L_∞ (right) errors of numerical solutions and their partial derivatives and gradients produced by the Multi-TransNet for the 3D diffusion interface problem (42) with the spatially varying diffusion coefficient (44).

and its gradient are illustrated in Fig. 23. It is clearly observed that for a fixed total number of hidden-layer neurons, all the results of the Multi-TransNet basically remain flat for different contrasts, except for a bump arising in the relative L_∞ errors of the solution and its gradient when $M = 2000$. With the doubling of the total number of hidden-layer neurons, the relative errors of both the Multi-TransNet solution and its gradient rapidly decrease.

6. Conclusions

In this paper, we propose a novel Multi-TransNet method, which integrates multiple distinct TransNets using the nonoverlapping domain decomposition approach, to solve various types of elliptic interface problems. Specifically, we design an efficient empirical formula-based prediction strategy for automatically selecting proper shape parameters for the subdomain TransNets, greatly reducing the tuning cost. Apart from that, the globally uniform distribution of the hidden-layer neurons across the entire domain is developed, with adaptive assignment of the number of hidden-layer neurons for each subdomain TransNet. During training, the weighting parameters are adaptively determined based on a normalization strategy, which balances the terms of the loss function and enhances the robustness. Extensive ablation studies and comparisons with some recently proposed neural network methods and traditional numerical techniques for solving classic 2D and 3D elliptic interface problems energetically demonstrate that the proposed Multi-TransNet method can offer striking performance in terms of accuracy, efficiency and robustness. To the best of our knowledge, this

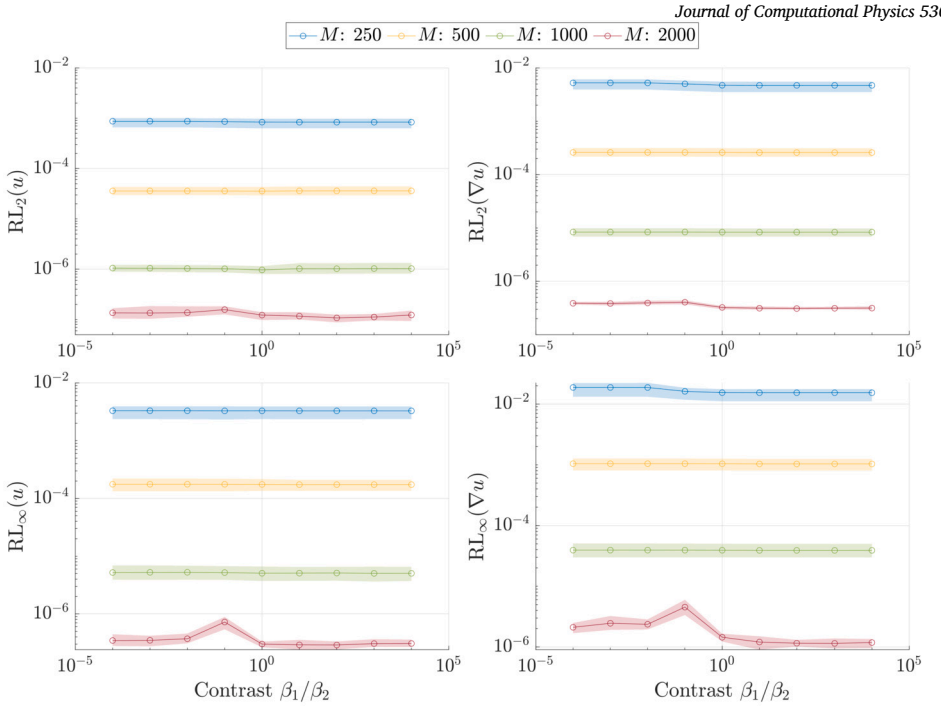

Fig. 23. The relative L_2 and L_∞ errors of the numerical solutions (left column) and their gradients (right column) produced by the Multi-TransNet with different total numbers of hidden-layer neurons ($M = 250, 500, 1000, 2000$) for the 3D diffusion interface problem (42) with the piecewise constant diffusion coefficient under different contrasts.

is the first time the TransNet technique has been extended to solve elliptic interface problems. Some potential future work includes: (1) improving the assembling and solving efficiency of the resulting least squares problem; (2) developing more effective approaches for generating the hidden-layer neurons based on specific target domains; (3) extending the present method to dynamic interface problems.

CRediT authorship contribution statement

Tianzheng Lu: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Visualization. **Lili Ju:** Writing – review & editing, Writing – original draft, Methodology, Conceptualization, Formal analysis, Supervision. **Liyong Zhu:** Writing – review & editing, Supervision, Methodology, Conceptualization, Formal analysis, Funding acquisition, Resources, Writing – original draft.

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

L. Zhu's work was partially supported by National Natural Science Foundation of China under grant number 11871091.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jcp.2025.113902.

Data availability

Data will be made available on request.

References

[1] T.Y. Hou, Z. Li, S. Osher, H. Zhao, A hybrid method for moving interface problems with application to the Hele–Shaw flow, J. Comput. Phys. 134 (2) (1997) 236–252.

[2] F. Gibou, D. Hyde, R. Fedkiw, Sharp interface approaches and deep learning techniques for multiphase flows, J. Comput. Phys. 380 (2019) 442–463.
[3] L. Greengard, M. Moura, On the numerical evaluation of electrostatic fields in composite materials, Acta Numer. 3 (1994) 379–410.
[4] D. Bochkov, T. Pollock, F. Gibou, A numerical method for sharp-interface simulations of multicomponent alloy solidification, J. Comput. Phys. 494 (2023) 112494.
[5] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (3) (1977) 220–252.
[6] K. Xia, X. Feng, Z. Chen, Y. Tong, G.-W. Wei, Multiscale geometric modeling of macromolecules I: Cartesian representation, J. Comput. Phys. 257 (2014) 912–936.
[7] R. Egan, F. Gibou, Fast and scalable algorithms for constructing solvent-excluded surfaces of large biomolecules, J. Comput. Phys. 374 (2018) 91–120.
[8] Z. Chen, Y. Xiao, L. Zhang, The adaptive immersed interface finite element method for elliptic and Maxwell interface problems, J. Comput. Phys. 228 (14) (2009) 5000–5019.
[9] M. Costabel, M. Dauge, S. Nicaise, Singularities of Maxwell interface problems, ESAIM: Math. Model. Numer. Anal. 33 (3) (1999) 627–649.
[10] I. Babuška, The finite element method for elliptic equations with discontinuous coefficients, Computing 5 (3) (1970) 207–213.
[11] Z. Chen, J. Zou, Finite element methods and their convergence for elliptic and parabolic interface problems, Numer. Math. 79 (2) (1998) 175–202.
[12] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[13] S.O. Unverdi, G. Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, J. Comput. Phys. 100 (1) (1992) 25–37.
[14] C.W. Hirt, B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1) (1981) 201–225.
[15] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations, J. Comput. Phys. 79 (1) (1988) 12–49.
[16] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (4) (1994) 1019–1044.
[17] Z. Li, K. Ito, The Immersed Interface Method: Numerical Solutions of PDEs Involving Interfaces and Irregular Domains, SIAM, 2006.
[18] R.P. Fedkiw, T. Aslam, B. Merriman, S. Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method), J. Comput. Phys. 152 (2) (1999) 457–492.
[19] X.-D. Liu, R.P. Fedkiw, M. Kang, A boundary condition capturing method for Poisson's equation on irregular domains, J. Comput. Phys. 160 (1) (2000) 151–178.
[20] R. Egan, F. Gibou, xGFM: recovering convergence of fluxes in the ghost fluid method, J. Comput. Phys. 409 (2020) 109351.
[21] F. Gibou, C. Min, R. Fedkiw, High resolution sharp computational methods for elliptic and parabolic problems in complex geometries, J. Sci. Comput. 54 (2013) 369–413.
[22] F. Gibou, R. Fedkiw, S. Osher, A review of level-set methods and some recent applications, J. Comput. Phys. 353 (2018) 82–109.
[23] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[24] B. Yu, W. E, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12.
[25] J. Sirignano, K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.
[26] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[27] Z. Cai, J. Chen, M. Liu, X. Liu, Deep least-squares methods: an unsupervised learning-based numerical method for solving elliptic PDEs, J. Comput. Phys. 420 (2020) 109707.
[28] Z. Wang, Z. Zhang, A mesh-free method for interface problems using the deep learning approach, J. Comput. Phys. 400 (2020) 108963.
[29] A.D. Jagtap, G.E. Karniadakis, Extended physics-informed neural networks (XPINNs): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, Commun. Comput. Phys. 28 (5) (2020).
[30] W. Li, X. Xiang, Y. Xu, Deep domain decomposition method: elliptic problems, in: Mathematical and Scientific Machine Learning, PMLR, 2020, pp. 269–286.
[31] J. Chen, X. Chi, Z. Yang, W. E, Bridging traditional and machine learning-based algorithms for solving PDEs: the random feature method, J. Mach. Learn. 1 (2022) 268–298.
[32] C. He, X. Hu, L. Mu, A mesh-free method using piecewise deep neural network for elliptic interface problems, J. Comput. Appl. Math. 412 (2022) 114358.
[33] S. Wu, B. Lu, INN: interfaced neural networks as an accessible meshless approach for solving interface PDE problems, J. Comput. Phys. 470 (2022) 111588.
[34] P.A. Mistani, S. Pakravan, R. Ilango, F. Gibou, JAX-DIPS: neural bootstrapping of finite discretization methods and application to elliptic problems with discontinuities, J. Comput. Phys. 493 (2023) 112480.
[35] D. Bochkov, F. Gibou, Solving elliptic interface problems with jump conditions on Cartesian grids, J. Comput. Phys. 407 (2020) 109269.
[36] G.-B. Huang, H.A. Babri, Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions, IEEE Trans. Neural Netw. 9 (1) (1998) 224–229.
[37] W.-F. Hu, T.-S. Lin, M.-C. Lai, A discontinuity capturing shallow neural network for elliptic interface problems, J. Comput. Phys. 469 (2022) 111576.
[38] Y.-H. Tseng, T.-S. Lin, W.-F. Hu, M.-C. Lai, A cusp-capturing PINN for elliptic interface problems, J. Comput. Phys. 491 (2023) 112359.
[39] S. Dong, Z. Li, Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations, Comput. Methods Appl. Mech. Eng. 387 (2021) 114129.
[40] Y. Li, F. Wang, Local randomized neural networks methods for interface problems, arXiv preprint, arXiv:2308.03087, 2023.
[41] X. Chi, J. Chen, Z. Yang, The random feature method for solving interface problems, Comput. Methods Appl. Mech. Eng. 420 (2024) 116719.
[42] Z. Zhang, F. Bao, L. Ju, G. Zhang, Transferable neural networks for partial differential equations, J. Sci. Comput. 99 (1) (2024) 1–25.
[43] B. Tang, Orthogonal array-based Latin hypercubes, J. Am. Stat. Assoc. 88 (424) (1993) 1392–1397.
[44] H. Dong, S. Li, W. Ying, Z. Zhao, Kernel-free boundary integral method for two-phase Stokes equations with discontinuous viscosity on staggered grids, J. Comput. Phys. 492 (2023) 112379.
[45] A. Guittet, M. Lepilliez, S. Tanguy, F. Gibou, Solving elliptic problems with discontinuities on irregular domains–the Voronoi interface method, J. Comput. Phys. 298 (2015) 747–765.
[46] X. Zhang, High order interface-penalty finite element methods for elasticity interface problems in 3D, Comput. Math. Appl. 114 (2022) 161–170.