Chen *et al. Advances in Aerodynamics* (2021) 3:36
https://doi.org/10.1186/s42774-021-00094-7

Advances in Aerodynamics

RESEARCH Open Access

Transfer learning for deep neural network-based partial differential equations solving

Xinhai Chen[1], Chunye Gong[1], Qian Wan[1], Liang Deng[2], Yunbo Wan[2], Yang Liu[2], Bo Chen[2] and Jie Liu[1]*

*Correspondence:
liujie@nudt.edu.cn
[1]Science and Technology on Parallel and Distributed Processing Laboratory, National University of Defense Technology, Changsha, China
Full list of author information is available at the end of the article

Abstract

Deep neural networks (DNNs) have recently shown great potential in solving partial differential equations (PDEs). The success of neural network-based surrogate models is attributed to their ability to learn a rich set of solution-related features. However, learning DNNs usually involves tedious training iterations to converge and requires a very large number of training data, which hinders the application of these models to complex physical contexts. To address this problem, we propose to apply the transfer learning approach to DNN-based PDE solving tasks. In our work, we create pairs of transfer experiments on Helmholtz and Navier-Stokes equations by constructing subtasks with different source terms and Reynolds numbers. We also conduct a series of experiments to investigate the degree of generality of the features between different equations. Our results demonstrate that despite differences in underlying PDE systems, the transfer methodology can lead to a significant improvement in the accuracy of the predicted solutions and achieve a maximum performance boost of 97.3% on widely used surrogate models.

Keywords: Deep neural network, Partial differential equation, Surrogate model, Transfer learning

1 Introduction

Neural network (NN) has been extensively studied as a surrogate model in the field of physics simulations for many years [1, 2]. Recent progress in deep learning offers a potential approach for the solution prediction of partial differential equations (PDEs) [3, 4]. Based on the universal approximation properties of the deep neural networks, pioneering works began to explore the possibility of building end-to-end solvers by means of the composition of hidden layers in a variety of network structures and activation functions [5–8]. These solvers consider the numerical simulation process as an unsupervised learning problem where the network model takes as input spatial and temporal coordinates and then predicts quantities of interests.

Physics-informed neural network (PINN) is one of the most commonly used DNN-based surrogate models [9, 10]. During the optimization phase, PINN embeds the

governing equations, as well as the initial/boundary conditions in the loss function as penalizing terms to guide the gradient descent direction. After suitable training, the network model is able to provide a nonlinear approximant for the underlying PDE systems. However, existing PINN-based surrogate models for PDE solving suffer from some inherent drawbacks. On the one hand, these models may not guarantee the desired prediction solution on the limited amount of training data, and tend to yield inaccurate results, especially for complex nonlinear PDEs [11, 12]. On the other hand, PINNs often require tedious training to converge, which are computationally cumbersome on modern CPUs or GPUs [13]. Moreover, due to the lack of generalizability, the trained network models are only applicable to the problem at hand and need to be re-trained for new problems or equation settings. The extensive re-training overhead limits their usefulness in engineering and optimal design context.

Transfer learning can be a powerful tool to enable efficient convergence during training. Generally, the transfer methodology is to train a base network and then transfer its learned features to the target network [14, 15]. By repurposing the knowledge and skills learned in previous tasks (domains), the network is able to produce a boost to prediction performance after fine-tuning to a new task (domain). Many studies have taken advantage of this fact to obtain state-of-the-art results in fields of computer vision applications, such as image classification and objects recognition [16–18]. However, the effect of transfer learning for DNN-based surrogate models has not been studied adequately.

In this paper, we aim not to optimize network designs, but rather to study the transfer ability of widely used surrogate models. We conduct a series of experiments on two PDE benchmarks using different models. Specifically, we first evaluate the transfer performance on PDEs with different equation settings. We also investigate the generality versus specificity of neurons in each layer of DNN-based surrogate models. Finally, we investigate the dependency of transfer results on the similarity of base and target tasks. The experimental results prove that initializing with transferred features can improve prediction accuracy after fine-tuning on a new task, which could be a generally useful technique for improving the prediction performance of surrogate models.

The rest of the paper is organized as follows. In Section 2, we present a brief overview of DNN-based surrogate models and introduce the methodology of transfer learning. Then, we conduct the transfer learning experiments. The prediction results are discussed in Section 3. In Section 4, we conclude the paper with a summary.

2 Related works

2.1 Deep neural network-based surrogate model

With the advances in neural network theory and learning algorithms, there has been much interest and work toward integrating deep neural networks into the traditional physics contexts [2, 19]. Recently, pioneering works began to solve nonlinear PDEs by utilizing the nonlinear approximate nature of deep neural networks. Raissi et al. [9, 10] first introduced the physics-informed neural network (PINN) for inferring the latent solution $u(\mathbf{x}, t)$ to a PDE system of the general form:

$$\frac{\partial^k u}{\partial t^k}(\boldsymbol{x},\ t)\ =\ \mathcal{D}[\,u(\boldsymbol{x},\ t)]\,, \qquad \boldsymbol{x}\in\Omega, \quad t\in[\,0,\ T\,]\,, \tag{1}$$

where the spatial domain $\Omega \in \mathbf{R}^d$, $\mathcal{D}$ is the differential operator. The PDE system is subject to the boundary condition,

$$\mathcal{B}[\,u(\boldsymbol{x},\ t)] \ =\ h(\boldsymbol{x},\ t), \qquad \boldsymbol{x} \in \partial\Omega, \tag{2}$$

where $\partial\Omega$ is the domain boundary, $\mathcal{B}$ is the differential operator for boundary conditions, and $h(\boldsymbol{x},\ t)$: $\mathbf{R}^{d+1} \mapsto \mathbf{R}$ is the given function.

In PINN, the solving process is formulated as a nonconvex optimization problem where an appropriate loss function is designed to optimize the predicted solution. Specifically, the governing equations, as well as the initial and boundary conditions, are embedded in the loss function as penalizing terms to guide the gradient descent direction:

$$Loss(\boldsymbol{x},\ t) \ =\ \left|\frac{\partial^k u}{\partial t^k}(\boldsymbol{x}_r,\ t) - \mathcal{D}[\,u(\boldsymbol{x}_r,\ t)]\right|^2 + |\mathcal{B}[\,u(\boldsymbol{x}_b,\ t)] - h(\boldsymbol{x}_b,\ t)|^2 \tag{3}$$

where $\boldsymbol{x}_r \in \Omega$ and $\boldsymbol{x}_b \in \partial\Omega$.

By minimizing this composite loss function, the trained network model is able to encode the underlying physical laws and work as a function approximator to provide the prediction $u(x, t)$. However, despite its simplicity, the original formulation of PINN often incurs difficulties in satisfying all equation residuals, thus leading to slow convergence or inaccurate prediction for many physical problems governed by complex nonlinear PDEs.

In order to alleviate the unstable approximation of original PINN, Wang et al. [11] studied the stiffness in the gradient flow dynamics of PINN and investigated the imbalance of the back-propagated gradients during gradient descent optimization. They proposed an adaptive learning rate algorithm along with a novel fully-connected neural architecture to solve the above deficiencies, resulting in two PINN-based variants: PINN-anneal and GP-PINN. They tested their network models across a range of problems in computational physics. The results show that, their methods outperform the original PINN and improve the accuracy by a factor of 50-100x. Following this line of research, many other similar surrogate solvers have been developed, such as DGM [20], hp-VPINNs [12], and NSFnets [13]. These models provide a series of promising results in many scientific applications such as computational fluid dynamics, electromagnetism, solid-state physics, and quantum physics [4].

2.2 Transfer learning

In computer vision, modern deep neural networks tend to learn first-layer features that resemble either Gabor filters or color blobs, and last-layer features that depend greatly on the underlying task [21, 22]. Based on this observation, transfer learning approaches are employed to transfer knowledge between the relevant base and target tasks to achieve efficient training convergence. As shown in Fig. 1, the usual transfer approach is to train a base network on a large dataset, and then transfer the learned features to a target network to be trained on a target dataset.

Yosinski et al. [14] experimentally studied the generality versus specificity of learned features in each network layer for different training tasks. Their results show that the appearance of the common (first-layer) features occurs not only to be specific to a particular image data or training set, but general in that they are applicable to different datasets. They also proved that the transfer process will tend to work well if the transferred features are not specific to the base task, but general to both base and target tasks. Many studies in

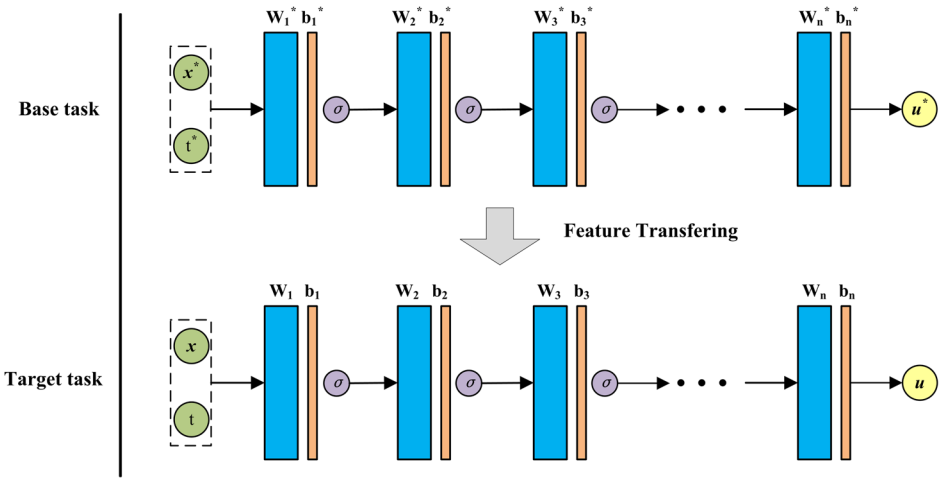


Fig. 1 Overview of the transfer learning approach. Instead of initializing the entire network randomly, the usual transfer approach is to train a base network on a large dataset and then copy (transfer) its first *n* layers to the first *n* layers of a target network. The remaining layers are randomly initialized and trained toward the target task

the image and natural language processing have taken advantage of this phenomenon to obtain state-of-the-art results. Examples of transfer learning include coping with image data with different lighting or background in similar tasks [16], overcoming the deficit of training samples for natural language domains [23, 24], and improving the classification or regression performance on unsupervised pseudo-tasks [15, 17].

Oquab et al. [21] studied the ability of a system to recognize and apply knowledge and skills learned in previous domains to novel domains. Their results collectively show that, by transferring image representations learned in previous domains, the underlying network pre-train the network produces a boost to prediction performance after fine-tuning to a new task, even with very different training objectives. Similar work was done by Sermanet et al. [25]. They proved that neural networks do indeed compute features that are fairly general, even across different data sources. The above results further emphasize the importance of studying the generality nature and extent of the learned features in different applications.

Overall, transfer learning can be a useful technique for improving the classification or regression performance in neural network training. However, there have not been any experimental results of DNN-based surrogate models for their transfer ability. Inspired by the tremendous success of transfer experiments in the areas of computer vision and image recognition, we apply the transfer learning approach to deep neural network-based surrogate models to investigate their transfer performance on different PDE systems.

3 Experimental results

In this study, we evaluate the performance of transfer learning on the neural network-based PDE solving tasks. In Section 3.1, we train DNN-based surrogate models on the Helmholtz equation and investigate the transfer performance in cases with different source terms. In Section 3.2, we conduct the transfer learning experiments on the Navier–Stokes equation with different Reynolds numbers. Finally, the transfer ability between two different equations is evaluated in Section 3.3.

For all experiments, we employ three widely used network models for PDE solving, including PINN, PINN with annealing, and GP-PINN [9–11]. We follow the network designs in these works and employ the original three-layer architectures in our solving tasks. Figure 2 shows an example of the PINN architecture. Due to the relatively shallow layers of the underlying networks, we do not consider the frozen-layer cases in this paper. During each training, we train the networks for 40000 epochs using the Adam optimizer in TensorFlow 1.15.0 [26]. The initial learning rate is set to 1e-03 and decays 0.95 every 1000 epochs. We use 128 random points per epoch as the training set and use 10000 uniform points as the test set at the end of the training.

3.1 Helmholtz equation

We first employ a classical benchmark, the two-dimensional Helmholtz equation [27], to investigate the performance of the transfer methodology. The Helmholtz equation is one of the fundamental PDEs in many scientific fields such as acoustics, electromagnetism, and elastic mechanics. It takes the form:

$$\Delta u(x,y) + k^2 u(x,y) = q(x,y), \qquad (x,y) \in \Omega, \tag{4}$$

$$u(x,y) = h(x,y), \qquad (x,y) \in \partial\Omega, \tag{5}$$

where Δ is the Laplace operator, $\Omega \in [0,1] \times [0,1]$, and $q(x,y)$ is a source term given by:

$$\begin{aligned} q(x,y) = &-(\alpha_1\pi)^2 \sin(\alpha_1\pi x)\sin(\alpha_2\pi y) - (\alpha_2\pi)^2 \sin(\alpha_1\pi x)\sin(\alpha_2\pi y) \\ &+ k^2 \sin(\alpha_1\pi x)\sin(\alpha_2\pi y). \end{aligned} \tag{6}$$

When k = 1, $h(x,y)$ is computed by the analytical solution given by:

$$u_{ref} = \sin(\alpha_1\pi x)\sin(\alpha_2\pi y). \tag{7}$$

During the training, we construct seven cases with different source terms for Helmholtz benchmarks: (1) $\alpha_1 = 1$, $\alpha_2 = 4$; (2) $\alpha_1 = 1$, $\alpha_2 = 6$; (3) $\alpha_1 = 2$, $\alpha_2 = 3$; (4) $\alpha_1 = 2$, $\alpha_2 = 4$; (5) $\alpha_1 = 3$, $\alpha_2 = 5$; (6) $\alpha_1 = 4$, $\alpha_2 = 5$; (7) $\alpha_1 = 4$, $\alpha_2 = 6$. To create the base and target subtasks, we set the case 1 ($\alpha_1 = 1, \alpha_2 = 4$) as the base subtask and the other six cases as target subtasks. Specifically, we first train all the subtasks using random initialization and obtain the original prediction results of six target subtasks. Then, we re-trained the six subtasks by repurposing the learned features of the base subtask to investigate the feature transfer ability on the Helmholtz benchmarks. When generalizing (transferring) to the other

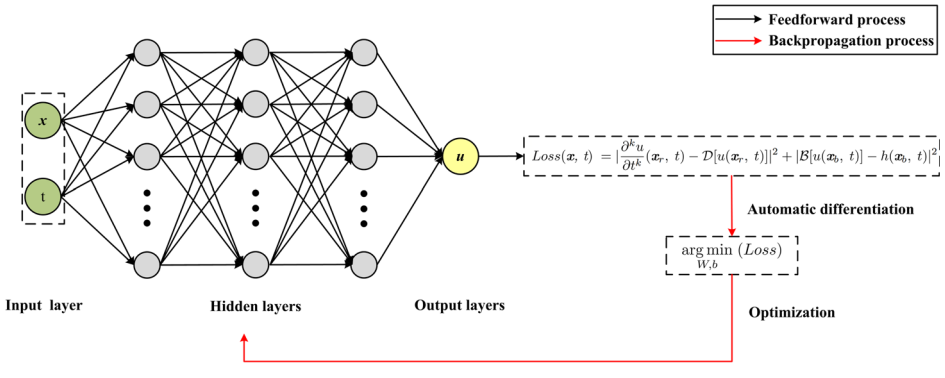


Fig. 2 The utilized network architecture of PINN

Table 1 Performance boost of PINN on the Helmholtz benchmarks using transfer learning

Subtask	Original	Transfer	Boost
case 2	1.04e+00	5.71e-01	45.1%
case 3	1.37e-01	7.34e-02	46.4%
case 4	5.48e-01	1.54e-01	71.9%
case 5	3.14e+00	5.55e-01	82.3%
case 6	4.95e+00	2.21e+00	55.4%
case 7	6.08e+00	5.02e+00	17.4%

cases, we would expect that the DNN-based solvers trained on top of the previous (base) subtask would perform better than random initialization since the underlying solution spaces of various settings are similar.

We summarize the prediction accuracy of different network models in Tables 1, 2 and 3. The prediction error is measured in the relative L^2 norm, which is defined as:

$$L^2 error = \frac{\|u_{ref} - u_{pred}\|_2}{\|u_{ref}\|_2}, \tag{8}$$

where u_{ref} and u_{pred} represent the reference/analytical solution and the predicted solution obtained by surrogate models.

The experimental results in Tables 1, 2 and 3 show that transferring base network variables and then fine-tuning them exhibit better performance than those trained directly on the target subtask. Compared with the original training, the transfer learning approaches achieve a maximum performance boost of 82.3% for PINN, 97.3% for PINN-anneal, and 55.6% for GP-PINN. Taking the PINN-anneal as an example, the network gives a relatively low prediction result (1.87e+00) in case 1. However, it is clear that the transfer model outperforms the original model by a large margin and achieves an L^2 error of 4.99e-02.

To better compare the performance boost in each target subtask, we also plot the visualization results of transfer learning experiments on the Helmholtz equation in Figs. 3 and 4. We can observe that repurposing the network variables of the base subtask is robust for physics-informed neural networks. Compared with the original training results depicted in the second column, we can clearly see the advantage of the transfer, that is, the prediction solution is more accurate in all cases. This is evidence that network models provide means to learn rich high-dimensional operator features transferrable to a variety of similar DNN-based PDE solving tasks. Moreover, we add an additional example to investigate whether transfer learning is likely to lead to worse results when the prediction accuracy of the base task is poor. In the new example, we take case 7 (Fig. 4c) as the base task and case 3 (Fig. 3b) as the target task. The experimental results demonstrate that transferring

Table 2 Performance boost of PINN-anneal on the Helmholtz benchmarks using transfer learning

Subtask	Original	Transfer	Boost
case 2	1.87e+00	4.99e-02	97.3%
case 3	2.95e-02	1.47e-02	50.2%
case 4	3.88e-02	2.49e-02	35.8%
case 5	4.17e-01	6.06e-02	85.5%
case 6	7.46e-01	2.18e-01	70.8%
case 7	1.47e+00	9.92e-01	32.5%

Table 3 Performance boost of GP-PINN on the Helmholtz benchmarks using transfer learning

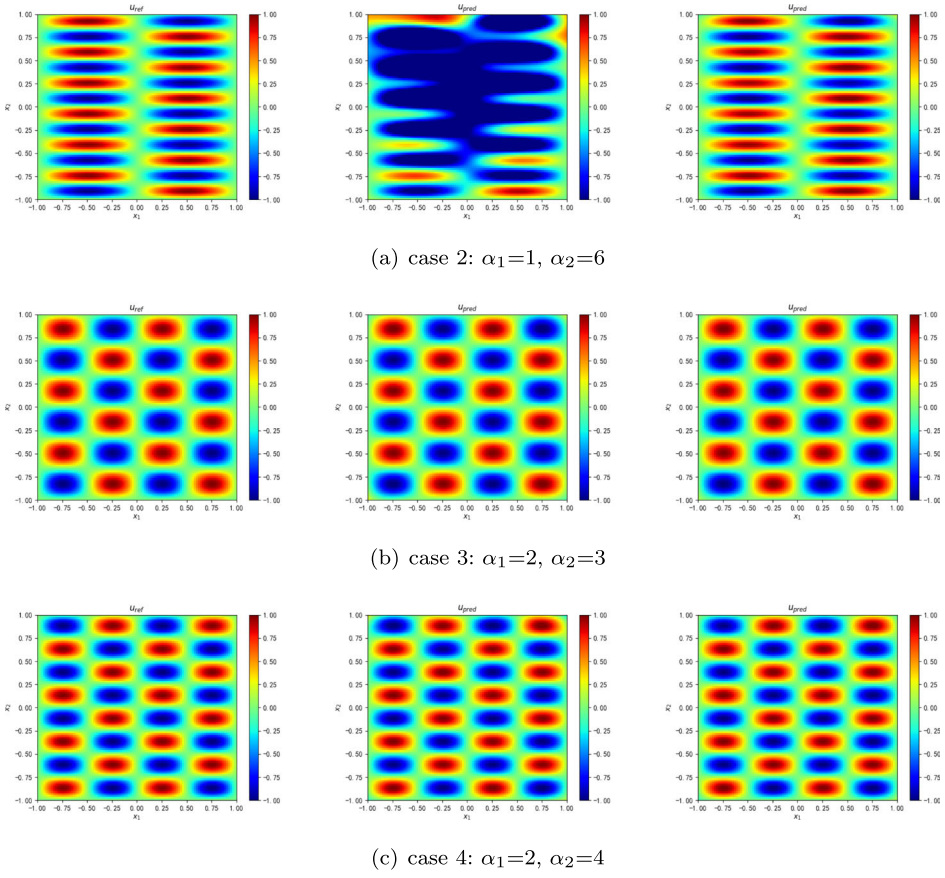
Subtask	Original	Transfer	Boost
case 2	9.01e-03	6.61e-03	26.6%
case 3	3.48e-03	2.55e-03	26.7%
case 4	9.20e-03	7.76e-03	15.7%
case 5	2.33e-02	2.24e-02	3.9%
case 6	3.04e-02	2.79e-02	8.2%
case 7	7.28e-02	3.23e-02	55.6%

features even from the base task with poor performance can be better than using random features. Taking PINN as an example, the original network achieves an L^2 error of 1.37e-01 in case 3, while yields 8.46e-02 using transfer learning.

3.2 Navier-Stokes equation

We now analyze the transfer performance of different models on the two-dimensional lid-driven cavity benchmarks (see Fig. 5). The steady-state flow in this problem is governed by the Navier-Stokes equation [19], which is defined as:

$$u(x,y)\cdot\nabla u(x,y)+\nabla p(x,y)-\frac{1}{Re}\Delta u(x,y)=0\quad (x,y)\in\Omega, \tag{9}$$

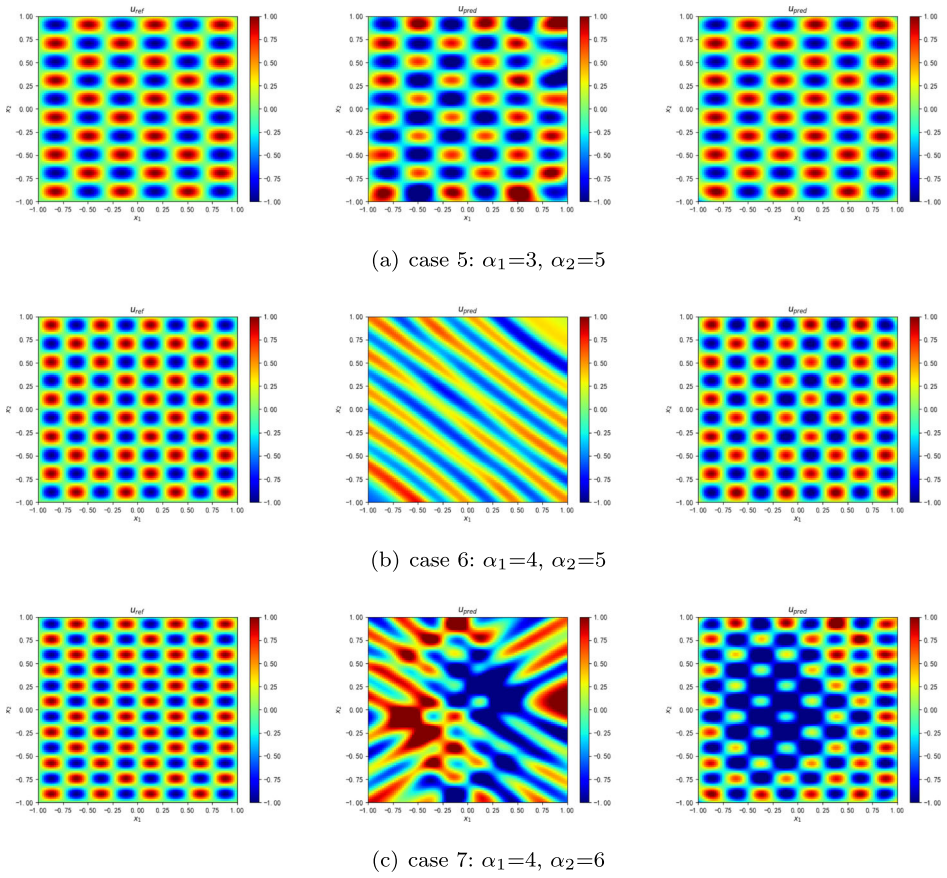


(a) case 2: α_1=1, α_2=6

(b) case 3: α_1=2, α_2=3

(c) case 4: α_1=2, α_2=4

Fig. 3 Prediction results of PINN-anneal on Helmholtz benchmarks (case 2-4). **Left**: reference solution. **Middle**: predicted solution using random initialization. **Right**: predicted solution using transfer learning



(a) case 5: α_1=3, α_2=5

(b) case 6: α_1=4, α_2=5

(c) case 7: α_1=4, α_2=6

Fig. 4 Prediction results of PINN-anneal on Helmholtz benchmarks (case 5-7). **Left**: reference solution. **Middle**: predicted solution using random initialization. **Right**: predicted solution using transfer learning

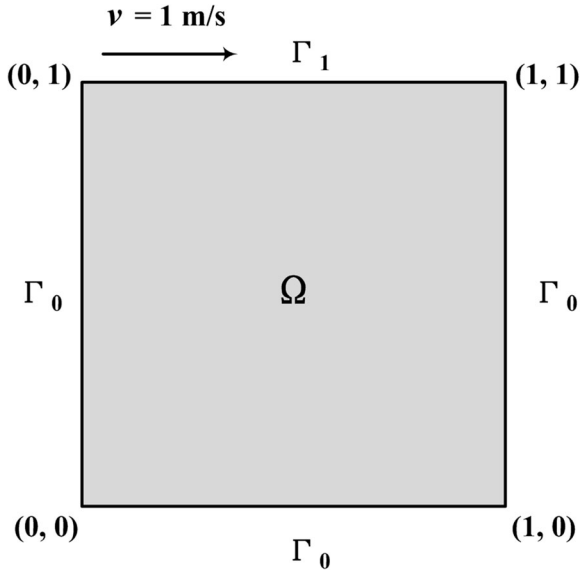


Fig. 5 Two-dimensional lid-driven cavity benchmark

Table 4 Performance boost of PINN on the Navier-Stokes equations using transfer learning

Subtask	Original	Transfer	Boost
$Re = 10$	1.12e-01	9.89e-02	11.7%
$Re = 200$	5.44e-01	2.90e-01	46.7%
$Re = 300$	6.07e-01	3.98e-01	34.4%
$Re = 400$	6.43e-01	5.23e-01	18.7%
$Re = 500$	6.73e-01	6.07e-01	9.8%

$$\nabla \cdot u(x, y) = 0 \qquad (x, y) \in \Omega, \tag{10}$$

$$v(x, y) = (1, 0) \qquad (x, y) \in \Gamma_1, \tag{11}$$

$$v(x, y) = (0, 0) \qquad (x, y) \in \Gamma_0, \tag{12}$$

where $\Omega \in [0, 1] \times [0, 1]$, Re is the Reynolds number of the flow, $u(x, y)$ is a velocity vector field, $p(x, y)$ is a scalar pressure field, and $v(x, y)$ denotes the velocity in the direction.

We create six subtasks with Reynolds numbers Re = 10, 100, 200, 300, 400, 500 to comprehensively study the transfer performance of surrogate models for solving the Navier-Stokes equations. Specifically, we set the second subtask ($Re = 100$) as the base subtask and the other five subtasks as target subtasks. During the training, we use vorticity-velocity (VV) formulation [11, 13] to construct the loss function and compare the prediction results with the reference solution obtained by OpenFOAM [28].

The original and transfer performance of PINN for Navier-Stokes equations is shown in Table 4. It is observed that, for different Reynolds numbers, the network benefits from transfer learning methods. In all subtasks, the transfer results outperform the original results and achieve an average performance boost of 22.4%. Similar results can be seen in Tables 5 and 6. Taking GP-PINN (Table 6) as an example, the original network achieves an L^2 error of 2.47e-01 for $Re = 200$, while yields 1.42e-01 using transfer learning. When $Re = 400$, GP-PINN obtains an L^2 error of 2.94e-01 in the transfer case, and the performance boost is 45.5%. In all cases, the maximum performance boost of PINN-anneal and GP-PINN is 48.1% and 50.0%, respectively.

In transfer learning tasks, the number of transfer layers has a significant impact on the prediction accuracy of surrogate models. Here, we conduct a series of experiments to evaluate the transferability of features across different subtasks. For this purpose, we test the performance boost when using transfer variables for the first n ($n \in [1, 4]$) layers for various Reynolds numbers and models. The experimental results are depicted in Fig. 6.

From Fig. 6, we find that initializing the underlying network with transferred features from almost any number of layers can produce a boost to the prediction accuracy of surrogate models after fine-tuning to a new Reynolds number. In most cases, transferring all

Table 5 Performance boost of PINN-anneal on the Navier-Stokes equations using transfer learning

Subtask	Original	Transfer	Boost
$Re = 10$	6.18e-02	5.52e-02	10.7%
$Re = 200$	4.91e-01	2.55e-01	48.1%
$Re = 300$	5.85e-01	3.70e-01	36.8%
$Re = 400$	6.56e-01	4.47e-01	31.9%
$Re = 500$	6.65e-01	5.04e-01	24.2%

Table 6 Performance boost of GP-PINN on the Navier-Stokes equations using transfer learning

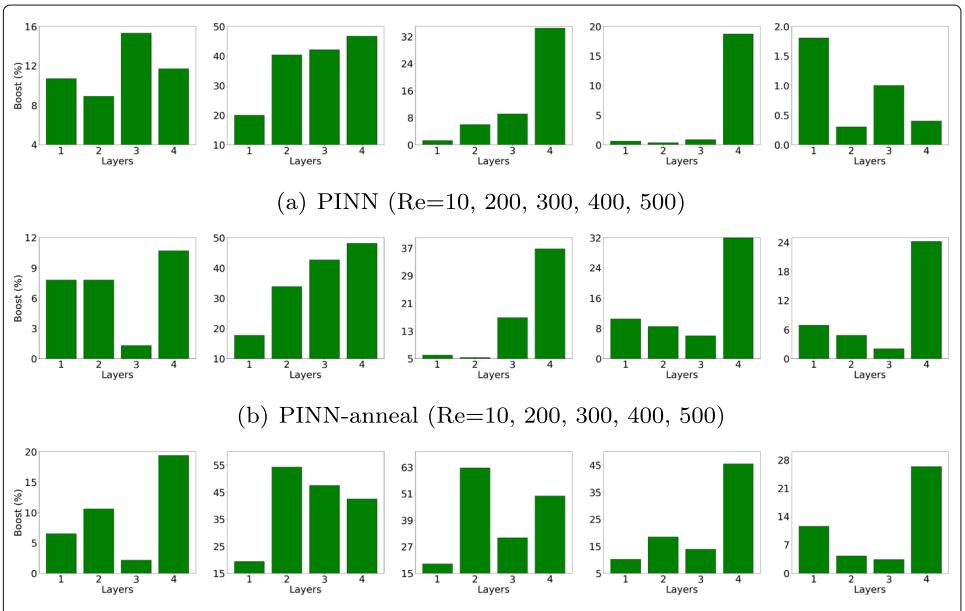
Subtask	Original	Transfer	Boost
$Re = 10$	4.63e-02	3.73e-02	19.4%
$Re = 200$	2.47e-01	1.42e-01	42.5%
$Re = 300$	4.92e-01	2.46e-01	50.0%
$Re = 400$	5.39e-01	2.94e-01	45.5%
$Re = 500$	5.59e-01	4.12e-01	26.3%

the layers achieves the best performance. Although the maximum boost may occur when $n = 1$ or 3, copying all the layers of the base subtask to the target training can still lead to considerable gains.

3.3 Transfer learning between different equations

In the last two sections, we conduct a series of transfer experiments by varying the Reynolds number or source terms of a PDE system. However, it is worth noting that there is a strong correlation between the above subtasks since the underlying solution spaces are similar. In order to study the transfer ability across dissimilar equations, we conduct experiments by transferring the features learned from the Navier-Stokes equation ($Re = 100$) to eight Helmholtz benchmarks. The eight target subtasks are: (1) $\alpha_1 = 1$, $\alpha_2 = 4$; (2) $\alpha_1 = 1$, $\alpha_2 = 6$; (3) $\alpha_1 = 2$, $\alpha_2 = 3$; (4) $\alpha_1 = 2$, $\alpha_2 = 4$; (5) $\alpha_1 = 3$, $\alpha_2 = 5$; (6) $\alpha_1 = 4$, $\alpha_2 = 5$; (7) $\alpha_1 = 4$, $\alpha_2 = 6$; (8) $\alpha_1 = 5$, $\alpha_2 = 6$.

Similarly, we test the prediction error using three different network models and the experimental results are shown in Tables 7, 8 and 9. We can see that transfer learning between different equations can still yield performance gains. Compared to random, untrained weights, PINN achieves an average performance boost of 50.6%. For PINN-anneal, the transferring learning approach delivers relatively large performance gains in



(a) PINN (Re=10, 200, 300, 400, 500)

(b) PINN-anneal (Re=10, 200, 300, 400, 500)

(c) GP-PINN (Re=10, 200, 300, 400, 500)

Fig. 6 The effect of different number of transfer layers on the prediction accuracy of the Navier-Stokes equations

Table 7 Performance boost of PINN on the Helmholtz benchmarks using transfer learning

Subtask	Original	Transfer	Boost
case 2	1.04e+00	6.03e-01	42.0%
case 3	1.37e-01	9.91e-02	27.7%
case 4	5.48e-01	1.70e-01	69.0%
case 5	3.14e+00	1.13e+00	64.0%
case 6	4.95e+00	3.22e+00	34.9%
case 7	6.08e+00	2.68e+00	55.9%
case 8	7.21e+00	2.83e+00	60.7%

Table 8 Performance boost of PINN-anneal on the Helmholtz benchmarks using transfer learning

Subtask	Original	Transfer	Boost
case 2	1.87e+00	1.21e-01	93.5%
case 3	2.95e-02	1.63e-02	44.7%
case 4	3.88e-02	2.21e-02	43.0%
case 5	4.17e-01	3.89e-01	6.7%
case 6	7.46e-01	7.02e-01	5.9%
case 7	1.47e+00	1.20e+00	18.4%
case 8	8.77e-01	8.07e-01	8.0%

Table 9 Performance boost of GP-PINN on the Helmholtz benchmarks using transfer learning

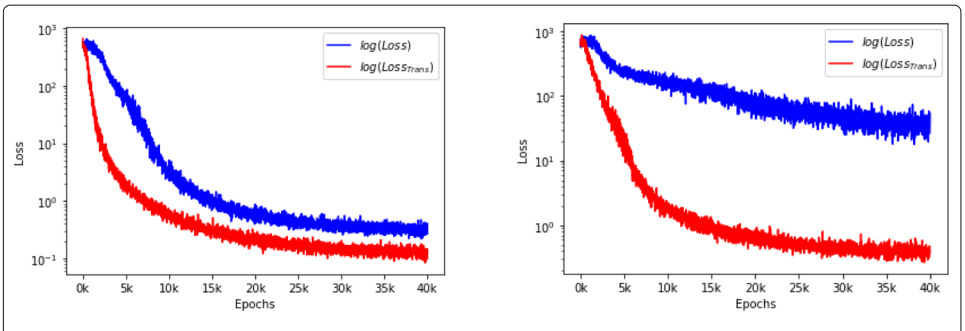
Subtask	Original	Transfer	Boost
case 2	9.01e-03	6.51e-03	27.7%
case 3	3.48e-03	3.30e-03	5.2%
case 4	9.20e-03	6.14e-03	33.3%
case 5	2.33e-02	1.89e-02	18.9%
case 6	3.04e-02	2.75e-02	9.5%
case 7	7.28e-02	3.71e-02	49.0%
case 8	8.59e-01	8.23e-02	90.4%

the first three cases, while providing acceptable performance gains in the last four cases. Similarly, the prediction error of GP-PINN can be largely reduced, yielding a performance boost ranging from 5.2%-90.4%.

To better analyze the transfer ability between different equations, we also plot the convergence of GP-PINN on cases 6 and 7 (see Fig. 7). From the variation curves of the loss value, we can observe that applying transferring learning is robust for PDE solution prediction. During the training phase, the value of the loss terms decreases as the learning rate decays with the increase in the epoch. However, the transferred model does a better job at fitting the governing equation and boundary conditions on Helmholtz benchmarks. Compared with training with random initialization, we can clearly see the advantage of transfer training, that is, it can help accelerate training convergence and yield more accurate results. Moreover, by comparing the prediction results obtained in Section 3.1, we find that the effectiveness of feature transfer declines as the base and target tasks become less similar. It is evident that the networks are able to achieve greater performance gains by using similar features (the same equation with different source terms) than by transferring variables obtained from another PDE system. However, transferring features even from different equations can be better than using random features.

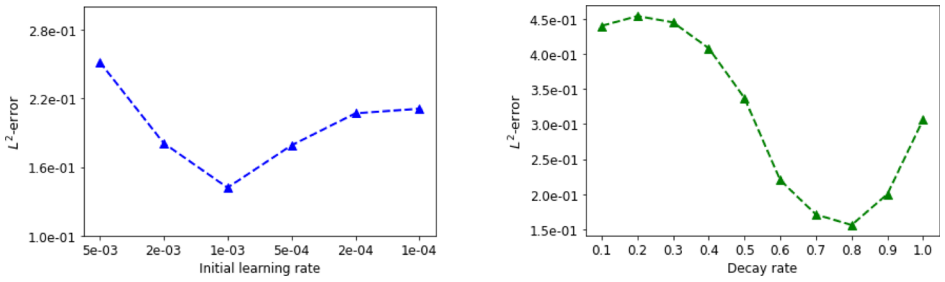
Different choices for hyperparameters can significantly affect the transfer performance for target subtasks. To establish a training methodology for DNN-based surrogate solvers, we analyze the effects of hyperparameters on the prediction accuracy of GP-PINN. The initial learning rate and decay rate are two of the most crucial hyperparameters for transfer learning. These two hyperparameters determine the length of optimization algorithm movement in the direction of the gradient. We test the effect of different learning rates and decay rates on the prediction accuracy of the Navier-Stokes equation ($Re = 200$) using GP-PINN .

The results in Fig. 8(a) show that overly large step size might have trouble converging, while small step size may get stuck in a local minima and provide suboptimal solutions. In our cases, GP-PINN achieves the best predictive performance when the initial learning rate is 1e-03. As for the decay rate, it is suggested that a relatively small decay rate is required to achieve a better performance, and the highest accuracy is achieved when the decay size is 0.8.



(a) case 6 (b) case 7

Fig. 7 The convergence of GP-PINN (on the log scale) on the Helmholtz equation. **Blue**: training with random initialization. **Red**: training with transfer learning



(a) The choice of the learning rate (b) The choice of the batch size

Fig. 8 A comparison of L^2 error for different initial learning rates (left) and decay rates (right)

4 Conclusion

Motivated by the tremendous success of transferred image representation in the areas of computer vision, we apply the transfer learning approach to neural network-based surrogate models to investigate the transfer ability on different PDE systems. Specifically, we analyze the transfer performance and show significant prediction improvements on the PDEs with different source terms and Reynolds numbers. We also quantify the degree of generality of features learned in each network layer. Moreover, we document that the transferability gap grows as the distance between the base and target tasks increases, but that transferring features even from distant tasks can be better than using random features. In future work, we will focus on applying the transfer methodology to more complex tasks in real-time analysis and optimization design applications. It is also interesting to study the feature transition from general to specific in more detail.

Acknowledgements
Not applicable.

Authors' contributions
These authors contributed equally to this work. All authors read and approved the final manuscript.

Funding
This work is supported by the National Numerical Windtunnel project (NNW2019ZT5-A10), and the National Key Research and Development Program of China (2018YFB0204301, 2017YFB0202104).

Availability of data and materials
The datasets used and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1] Science and Technology on Parallel and Distributed Processing Laboratory, National University of Defense Technology, Changsha, China. [2] China Aerodynamics Research and Development Center, Mianyang, China.

Received: 9 September 2021 Accepted: 20 October 2021
Published online: 08 December 2021

References

1. Chen X, Liu J, Pang Y, Chen J, Chi L, Gong C (2020) Developing a new mesh quality evaluation method based on convolutional neural network. Eng Appl Comput Fluid Mech 14(1):391–400
2. Chen X, Liu J, Gong C, Li S, Pang Y, Chen B (2021) MVE-Net: An automatic 3-D structured mesh validity evaluation framework using deep neural networks. Comput Aided Des 141:103104. https://doi.org/10.1016/j.cad.2021.103104

3. Chen T, Chen H (1995) Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. IEEE Trans Neural Netw 6(4):911–917
4. Brink AR, Najera-Flores DA, Martinez C (2020) The neural network collocation method for solving partial differential equations. Neural Comput Appl 33:5591–5608. https://doi.org/10.1007/s00521-020-05340-5
5. Fang Z (2021) A high-efficient hybrid physics-informed neural networks based on convolutional neural network. IEEE Trans Neural Netw Learn Syst 99:1–13. https://doi.org/10.1109/TNNLS.2021.3070878
6. Sun L, Gao H, Pan S, Wang J-X (2020) Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. Comput Methods Appl Mech Eng 361:112732. https://doi.org/10.1016/j.cma.2019.112732
7. Jagtap AD, Kawaguchi K, Karniadakis GE (2020) Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. J Comput Phys 404:109136. https://doi.org/10.1016/j.jcp.2019.109136
8. Yang L, Zhang D, Karniadakis GE (2020) Physics-informed generative adversarial networks for stochastic differential equations. SIAM J Sci Comput 42(1):292–317. https://doi.org/10.1137/18M1225409
9. Raissi M, Perdikaris P, Karniadakis GE (2019) Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 378:686–707. https://doi.org/10.1016/j.jcp.2018.10.045
10. Raissi M, Yazdani A, Karniadakis GE (2020) Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science 367(6481):1026–1030. https://doi.org/10.1126/science.aaw4741
11. Wang S, Teng Y, Perdikaris P (2020) Understanding and mitigating gradient pathologies in physics-informed neural networks. arXiv:2001.04536. http://arxiv.org/abs/2001.04536
12. Kharazmi E, Zhang Z, Karniadakis GEM (2021) hp-VPINNs: Variational physics-informed neural networks with domain decomposition. Comput Methods Appl Mech Eng 374:113547. https://doi.org/10.1016/j.cma.2020.113547
13. Jin X, Cai S, Li H, Karniadakis GE (2021) NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. J Comput Phys 426:109951. https://doi.org/10.1016/j.jcp.2020.109951
14. Yosinski J, Clune J, Bengio Y, Lipson H (2014) How transferable are features in deep neural networks? In: Advances in Neural Information Processing Systems 27 (NIPS' 14), Montréal, Canada, 8-13 December 2014
15. Saito K, Watanabe K, Ushiku Y, Harada T (2018) Maximum classifier discrepancy for unsupervised domain adaptation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp 3723–3732. https://doi.org/10.1109/CVPR.2018.00392
16. Chambino LL, Silva JS, Bernardino A (2021) Multispectral face recognition using transfer learning with adaptation of domain specific units. Sensors 21(13):4520
17. Aich S, Yamazaki M, Taniguchi Y, Stavness I (2020) Multi-scale weight sharing network for image recognition. Pattern Recogn Lett 131:348–354. https://doi.org/10.1016/j.patrec.2020.01.011
18. Ganin Y, Ustinova E, Ajakan H, Germain P, Larochelle H, Laviolette F, Marchand M, Lempitsky V (2016) Domain-adversarial training of neural networks. J Mach Learn Res 17(1):2096–2030
19. Arthurs CJ, King AP (2021) Active training of physics-informed neural networks to aggregate and interpolate parametric solutions to the Navier-Stokes equations. J Comput Phys 438:110364. https://doi.org/10.1016/j.jcp.2021.110364
20. Sirignano J, Spiliopoulos K (2018) DGM: A deep learning algorithm for solving partial differential equations. J Comput Phys 375:1339–1364. https://doi.org/10.1016/j.jcp.2018.08.029
21. Oquab M, Bottou L, Laptev I, Sivic J (2014) Learning and transferring mid-level image representations using convolutional neural networks. In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14). IEEE Computer Society, USA. pp 1717–1724. https://doi.org/10.1109/CVPR.2014.222
22. Ammar HB, Eaton E, Luna JM, Ruvolo P (2015) Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning. In: Yang Q, Wooldridge (eds). Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15). AAAI Press, Buenos Aires. pp 3345–3351
23. Keneshloo Y, Ramakrishnan N, Reddy CK (2019) Deep transfer reinforcement learning for text summarization. In: Proceedings of the 2019 SIAM International Conference on Data Mining. SIAM, USA. pp 675–683
24. Chen YS, Chiang SW, Wu ML (2021) A few-shot transfer learning approach using text-label embedding with legal attributes for law article prediction. Appl Intell. https://doi.org/10.1007/s10489-021-02516-x
25. Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, LeCun Y (2014) Overfeat: Integrated recognition, localization and detection using convolutional networks. In: Paper presented at 2nd International Conference on Learning Representations (ICLR 2014), Banff, Canada, 14-16 April 2014
26. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M, Levenberg J, Monga R, Moore S, Murray DG, Steiner B, Tucker P, Vasudevan V, Warden P, Wicke M, Yu Y, Zheng X (2016) Tensorflow: A system for large-scale machine learning. In: Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16). USENIX Association, Savannah. pp 265–283
27. Babuška I, Ihlenburg F, Paik ET, Sauter SA (1995) A generalized finite element method for solving the Helmholtz equation in two dimensions with minimal pollution. Comput Methods Appl Mech Eng 128:325–359. https://doi.org/10.1016/0045-7825(95)00890-X
28. Jasak H, Jemcov A, Tukovic Z (2007) OpenFOAM: A C++ library for complex physics simulations. In: Paper presented at International Workshop on Coupled Methods in Numerical Dynamics, IUC, Dubrovnik, Croatia, 19-21 September 2007

Publisher's Note