# VgEmPINN: An improved model for solving partial differential equations using deep neural networks

# Abstract

Partial differential equations (PDEs) are the basis of modern scientific theoretical systems, and solving PDEs is the basic problem of scientific computing. When numerical methods are used to solve PDEs, the solution domain needs to mesh, which is easy to result in the curse of dimensionality when solving high-dimensional PDEs. Based on deep learning technology, Physics-Informed Neural Networks (PINNs) can solve PDEs using neural networks embedded with physical information. When solving high-dimensional PDEs, this type of neural network has a great solving ability. This paper focuses on researching and optimizing this type of neural network to solve PDEs efficiently. Aiming at the problem of low accuracy in solving PDEs of PINN, we propose an improved unsupervised learning method for solving PDEs: VgEmPINN. To validate its correctness and robustness, the VgEmPINN method is then applied to solve the two-dimensional Poisson’s equation and Burgers’ equation. The experimental results show that the VgEmPINN method can improve the accuracy of solving equations by two orders of magnitude when compared with the basic PINN method, which suggests that it has important application value for solving PDEs using artificial intelligence technology.

# Introduction

Artificial intelligence, a new technology driver, is driving changes in various industries. With the rapid growth of available data and computing resources, AI techniques such as deep learning [1] have been widely used in a variety of disciplines such as image recognition and natural language processing, which has achieved transformative results with the emergence of classical approaches such as convolutional neural network (CNN) [2] and transformer [3]; however, its use in solving partial differential equations (PDEs) has emerged only recently.

Since the 1970s, various mesh-based numerical methods, such as finite difference, finite element, and finite volume methods [4], have been developed to solve PDE systems. These numerical methods require the computational domain of the PDE to mesh into grids and then solve the resulting linear or nonlinear system of PDEs for the unknown state variables at the nodes of the grid. The smaller the grid nodes are, the greater the resolution and the higher the accuracy of the solution. However, when solving PDEs using numerical methods, the number of grid nodes increases exponentially as the dimensionality of the equations increases, and the curse of dimension occurs, making it particularly difficult to solve high-dimensional PDEs using traditional numerical methods. In addition, solving inverse problems (for inferring material properties in functional materials or discovering missing physics in reactive transport) is prohibitively expensive and requires complex formulations.

Instead of discretizing the computational domain into mesh grids and then iteratively solving the system of PDEs on each subdomain, an alternative method using deep learning technology called physics-informed neural networks (PINNs) [5], can solve the forward and inverse problems of PDEs successfully via embedding the PDE into the loss of the neural network using automatic differentiation (AD) [6]. The idea of PINNs first originated from a work by Lagaris et al. in 1998 [7], but was not given much attention at that time. When studying Gaussian processes [8-9], Raissi et al. applied prior knowledge in the field of physics and achieved good performance, which was called physics-informed learning. Thanks to the automatic differentiation technique, the idea of physics-informed learning was able to be applied in neural networks: add a PDE residual term and boundary conditions terms into the loss function of the neural network, which is applied as prior knowledge, and the differential terms in the PDE are computed using AD, finally, the loss function can be optimized.

Since then, a series of related studies emerged, which improved the PINN method in different aspects such as activation function [10-11], a variational form of PDEs [12-13], gradient-enhanced [14], and PDE domain decomposition [15], all of them achieved better solution accuracy.

The question is whether the improvements of PINNs from different aspects can be applied to the solution of all PDEs and whether the accuracy and efficiency can be improved. Some researchers have used the technique of loss landscape [16~17] to explore the interpretability of the model. This visualization technique allows an intuitive understanding of how the loss function affects the generalization performance and trainability of a neural network.

In this work, we first develop an unsupervised method for solving PDEs: EmPINN. The core of EmPINN lies in the expanding layer. Before the training data enter the neural network, a dimensional expansion operation is performed in the expanding layer to ascend the low-dimensional solution domain space into a high-dimensional space. Besides, the neural network is changed from an ordinary multi-layer perception (MLP) to a modified MLP [18] with residual connectivity. After dimensional expansion and residual connectivity, EmPINN is able to map the initial low-dimensional solution space to a higher-dimensional feature space. It’s known that the neural network is best at handling high-dimensional problems, so EmPINN can make greater use of the fitting ability of the neural network.

Then we propose an improved unsupervised method for solving PDEs: VgEmPINN to address the problem of the low accuracy of PINN methods for solving PDEs. VgEmPINN combines the ideas of VPINN [13] and gPINN [14] based on EmPINN, which make full use of the variational information and the gradient information of the PDE residual. VPINN adds the integral form of PDE residuals to the loss function, gPINN adds the differential form of PDE residuals to the loss function, and VgEmPINN combines them so that the loss function contains the loss term of boundary conditions, the loss term of PDE residuals, the loss term of the integral form of PDE residuals and the loss term of the differential form of PDE residuals, and assign different weights to each loss term to make the model training more focused and optimize the training process of the model. The VgEmPINN method can not only enhance the application of prior knowledge to a greater extent but also make more use of the fitting ability of neural networks. The output results of the neural network can satisfy the equation constraints and be more consistent with the underlying physical laws. Experiments using the VgEmPINN method to solve Poisson’s equation and Burgers’ equation show that it can improve the accuracy of solving the equations by two orders of magnitude. This has important implications for solving PDE problems using artificial intelligence techniques.

The paper is organized as follows: in “Methods”, we first introduce the problem setup and provide a recap of PINN for solving PDE. Next, we develop our unsupervised methods: EmPINN and VgEmPINN. The proposed method is then applied to different PDEs to validate its correctness and robustness. We systematically compare the performance of PINN, gPINN, VPINN, hp-VPINN, EmPINN, and VgEmPINN in “Results and discussion”. Finally, conclusions and future work are drawn in “Conclusion”. All the codes in this article are based on Python 3.7.4 and Tensorflow 1.14.0. All numerical experiments reported here are running on the laptop with the 10th Gen Intel(R) Core(TM) i7-10510U @ 1.80 GHz processor and 16.0 GB of memory.

# Methods

# Problem Setup

PDEs are the basis of modern scientific theoretical systems, and solving PDEs is a fundamental problem in scientific computing. In this paper, we consider the general form of a system of nonlinear PDEs as follows:

where is the solution of the PDE, and are nonlinear operators. For example, in Burgers’ equation, , . There are three main types of boundary conditions for PDEs. The Dirichlet boundary conditions give the value of the unknown function on the boundary, the Neumann boundary conditions give the value of the derivative of the unknown function on the boundary, and the Robin boundary conditions give a linear combination of the value of the function and the value of the derivative of the unknown function on the boundary. The boundary condition of the mentioned Burgers' equation is a Neumann boundary condition. There are two types of problems for solving PDEs: forward problems and inverse problems. In this paper, we study the forward problem, which is to solve the numerical solution of when the parameters of the equation are all known.

To evaluate the accuracy of different solving methods, we use the relative -error criterion, which is defined as:

# Neural Networks

In this section, we aim to introduce neural networks. Recall the mathematical theorem of neural networks as universal function approximators [19]. For a continuous function defined on and is a nonlinear activation function, then can be approximated by the following expression:

where is a sufficiently small positive number, , and are real numbers, i.e., the weights and biases in the neural network, is the value of at points , and the hidden layer of the neural network has neurons. Equation (2.4) ensures that the error between the output of the neural network and the function to be approximated is within a sufficiently small range, i.e., .

Fig 2-1 shows the structure of the single hidden layer neural network of Equation (2.4). The input layer is the value of at input nodes, and the hidden layer performs linear operations on the input values using the weights and bias followed by nonlinear activation through the activation function . Finally, the results of the hidden layer nodes are multiplied with the weights and summed up as the output . Considering the trainability of neural networks, using only single hidden layer neural networks is less practical than using neural networks containing multiple hidden layers, i.e., deep neural networks. Therefore, all the neural networks used in this paper are deep neural networks.

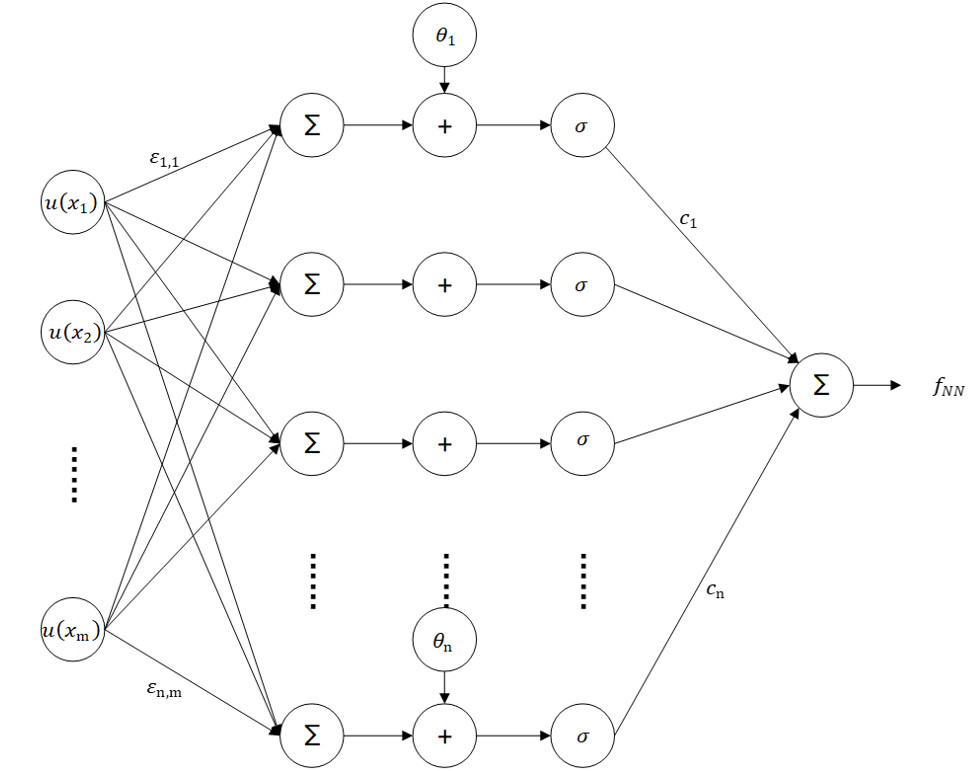


Fig 2-1 neural network as universal function approximators

# The PINN method

This subsection introduces the principle of PINN using the Burgers' equation given in Equation (2.1) as an example.

First, a deep neural network is defined to approximate the solution function . The input is the spatio-temporal coordinates and the output is the solution of Burgers' equation. The neural network is then defined to approximate the PDE. The loss function is defined as follows:

in the formula is the set of initial boundary points, is obtained by the Latin hypercube sampling (LHS) strategy [20] in the solution domain, is the number of initial boundary points, is the number of sampled points, and is the value of the function that satisfies the initial and boundary value conditions.

PINN embeds the PDE into the loss function of the neural network, and the calculation of it can be performed using the automatic differentiation technique, the principle of which is shown in Fig 2-2. The resulting trained deep neural network can encode the underlying physical laws as a priori information, and finally obtains the numerical solution of the PDE. Compared with ordinary neural networks, PINN has the advantages of not requiring a large amount of sample data for training, being an unsupervised method, and generalizing well.

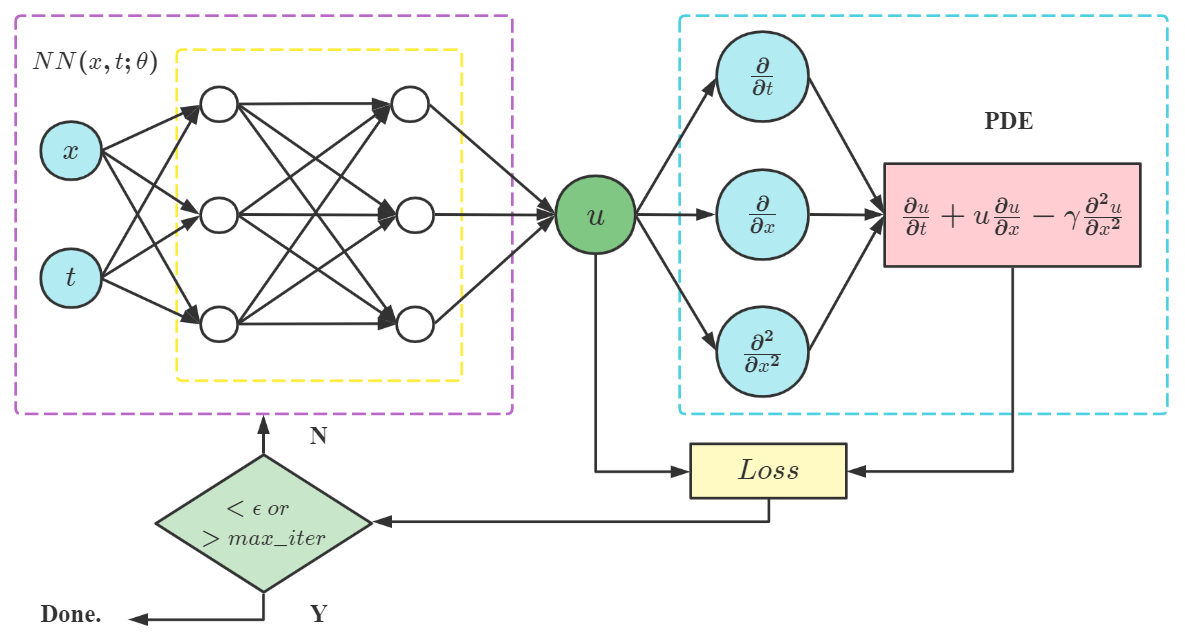


Fig 2-2 schematic diagram of PINN

# The EmPINN method

Following the original works of PINN, we propose an improved method called EmPINN to address the problem of the low accuracy of PINN methods for solving PDEs.

First, we introduce an expanding layer [21] that extends the dimensionality of the input features, which extends the dimensionality of the input tuple into . This layer defines a mapping from the input layer to the output :

where computes the square of the element-wise.

After this feature transformation, the solution domain is mapped from a two-dimensional input function space to new four-dimensional function space. The PDE contains analytic solutions such as Poisson’s equation, its analytic solutions usually contain higher power terms such as . Therefore, the neural network is trained and optimized in the new function space, and it is easier to fit the solution of PDE. Second, this paper uses modified MLP [22] instead of MLP used in PINN, which has the advantage of enhancing the hidden states with residual connections. The network structure of the modified MLP is shown in Fig 2-3.

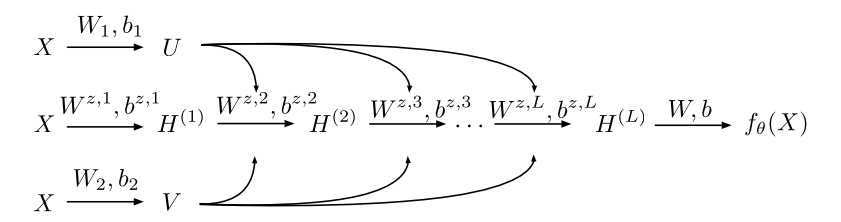


Fig 2-3 network structure of modified MLP [23]

The forward pass of modified MLP is defined as follows:

where denotes point-wise multiplication, denotes an activation function.

By combining the idea of extended dimensionality with residual connections, the initial two-dimensional solution domain space can be mapped to a higher-dimensional feature space, and neural networks are best at handling high-dimensional problems, so EmPINN can make greater use of the fitting ability of neural networks and improve the accuracy of PINN in solving PDEs.

EmPINN schematic diagram is shown in Fi 2-4.

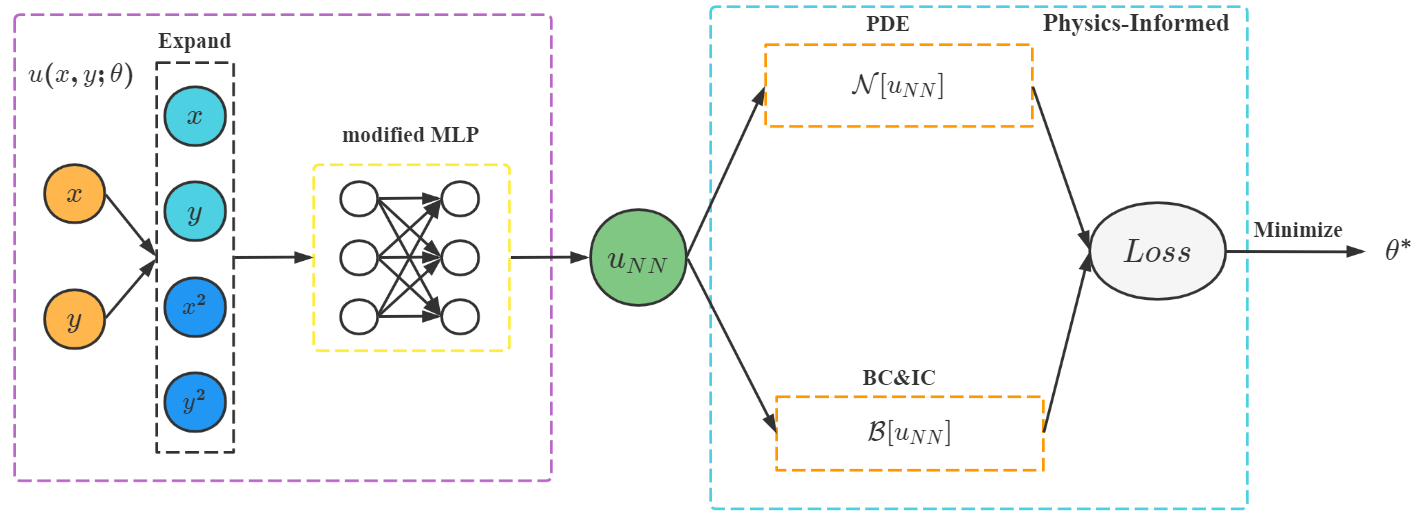


Fig 2-4 schematic diagram of EmPINN

The definition of the loss function for the EmPINN method is identical to that of PINN, i.e., Equation (2.5). The difference is that the deep neural network is defined in PINN to approximate the solution function , and the defined above is used in EmPINN to approximate the solution function . After that the same optimization process was performed for both.

# The VgEmPINN method

EmPINN changes the original network structure of PINN. While existing methods such as gPINN and hp-VPINN improve PINN in terms of loss functions. They add differential and integral forms of the PDE to the loss function. Different improvements in network structure and loss function are possible to combine, so this paper proposes the VgEmPINN method based on the already proposed EmPINN, combined with the gradient-enhanced method and variational method and the idea of domain decomposition. Using the powerful fitting ability of neural networks in EmPINN, and the ability to make full use of the physical information in gPINN and hp-VPINN, VgEmPINN is more capable of improving the accuracy of PINN for solving PDEs. The schematic diagram of VgEmPINN is shown in Fig 2-5.

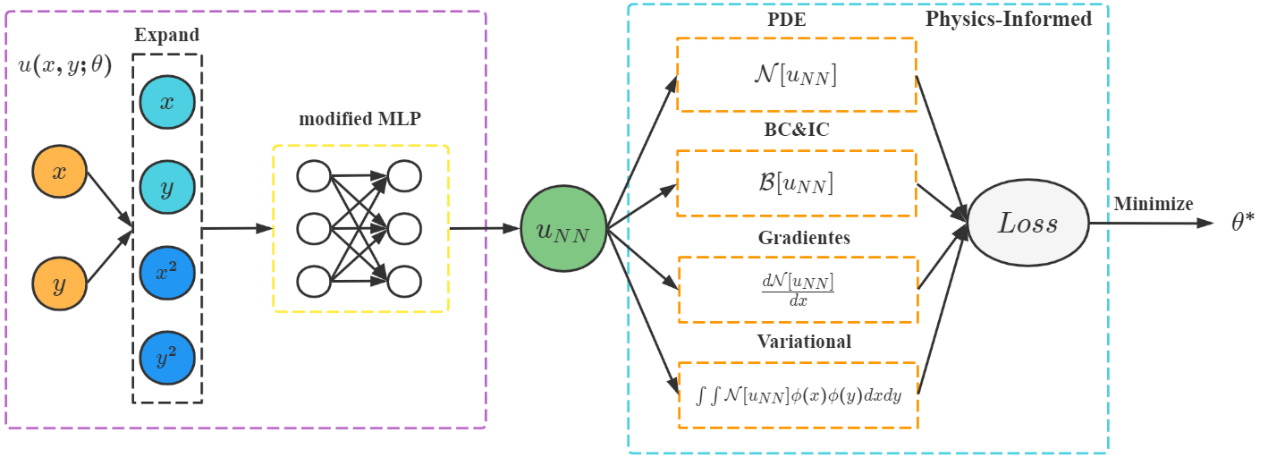


Fig 2-5 schematic diagram of VgEmPINN

Taking Poisson’s Equation as an example, the loss function is defined as follows:

is the set of boundary points, and are the set of randomly sampled points in the solution domain, and is the set of quadrature points. is the number of boundary points, is the number of sampled points, and is the value of the function satisfying the boundary condition. and are the number of domains decomposed in and dimensions, respectively. and are the number of subdomains in dimension and dimension of the test function. And each loss term of the loss function is assigned a different weight in such a way that the model can be better trained [23]. is the weight of each loss term, is the weight of the boundary loss term, is the weight of the PDE residual loss term, and is the weight of the integral form of the PDE residual loss term.

# Results and Discussion

In this section, we study and compare the performance of VgEmPINN with some other widely used DNN-based methods for PDE solving, including PINN, gPINN, and hp-VPINN.

In all examples for solving PDEs, we use the tanh as the activation function. In order to minimize the loss function, we employ the Adam optimizer and L-BFGS-B [24] method to optimize the parameter of the neural network. We first apply the Adam optimizer for stochastic gradient descent training and then employ the L-BFGS-B optimizer to finetune the results. L-BFGS-B is a limited-memory quasi-Newton optimizer for bound-constrained optimization. It is known to work very well at escaping from local optima during network training and requires little tuning. The other hyperparameters for each example are listed in Table 1. In our work, the learning rate is , and the random seed of TensorFlow [25] and Numpy [26] is set to 1234 during training to ensure the reproducibility of the experimental results.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Section number and problem | Depth | Width |  |  | Iterations |
| 3.1 Poisson’s Equation | 3 | 20 | 50 | 80 | 500 |
| 3.2 Burgers’ Equation | 5 | 40 | 5000 | 100 | 10000 |

Table 1. Hyperparameters for the equations tested in this study.

# Poisson’s equation

Poisson’s equation is a common PDE in mathematics for statics, mechanical engineering, and theoretical physics, and one of the three fundamental equations for microelectronic devices, which is widely used. In this paper, we consider solving the two-dimensional Poisson’s equation, which has the following equation form:

It is satisfying the Dirichlet boundary condition.

In this paper, for ease of verification, we solve the homogeneous two-dimensional Poisson’s equation that holds constant over the entire solution domain.

As a result, the equation can be solved analytically as follows.

The loss function is

where , , and are defined in Eqs. (2.12), (2.13), (2.14) and (2.15), respectively.

Fig 3-1 depicts the convergence of VgEmPINN for solving Poisson’s equation.

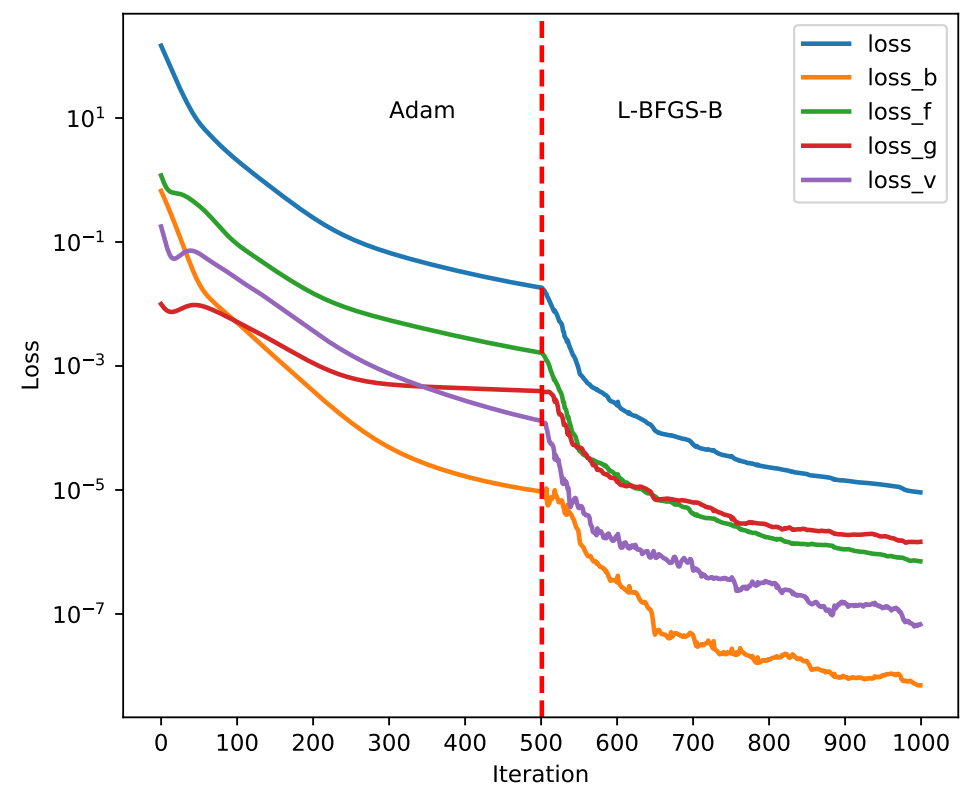


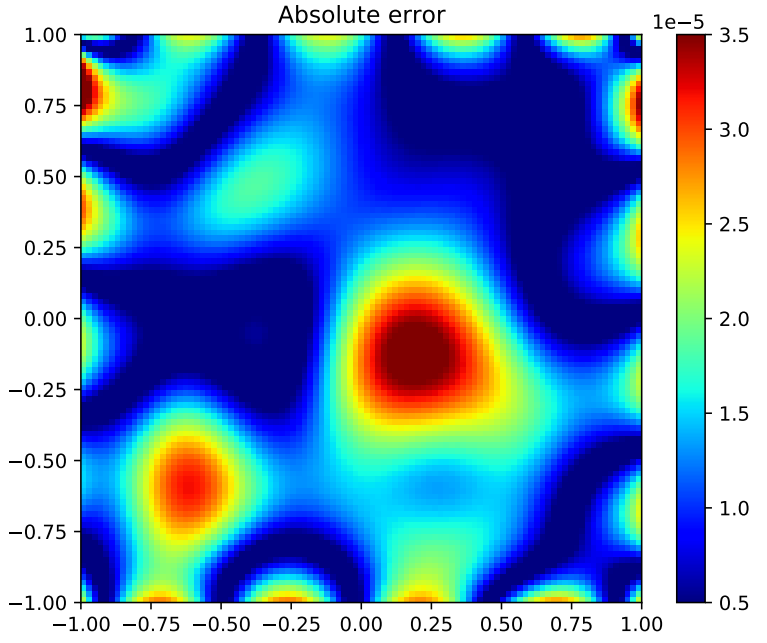
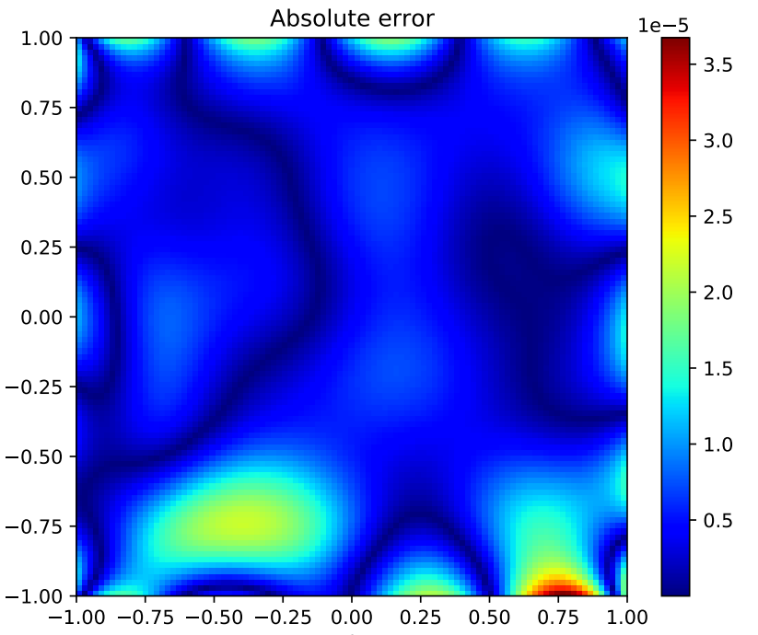
Fig 3-1 The convergence of VgEmPINN (on the log scale) on Poisson’s equation. The Adam optimizer is used before the vertical dashed line, and the L-BFGS-B optimizer is used afterward.

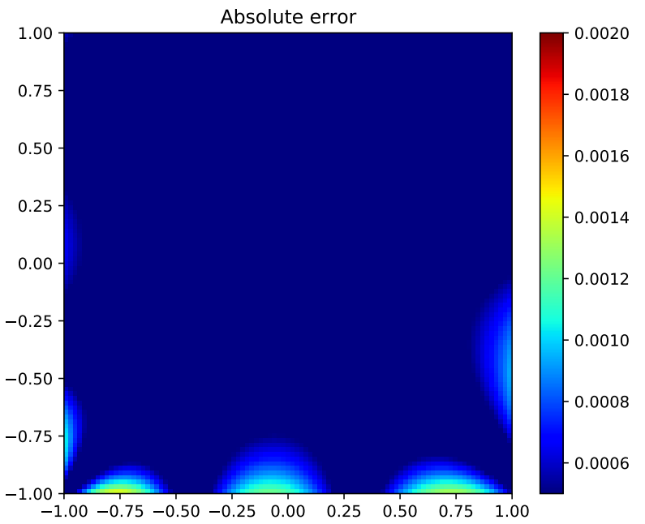
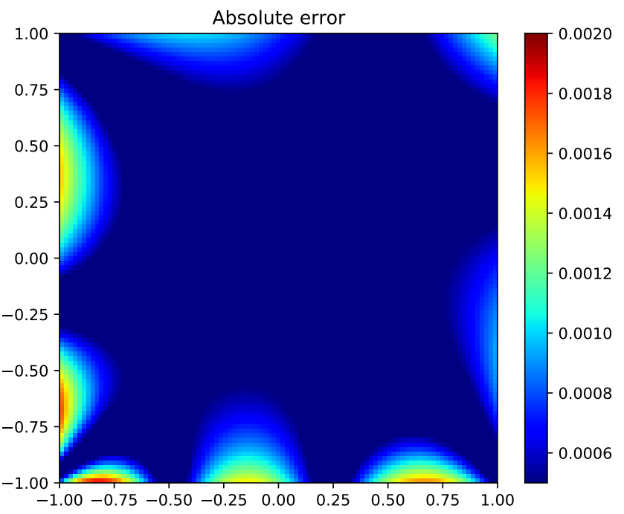
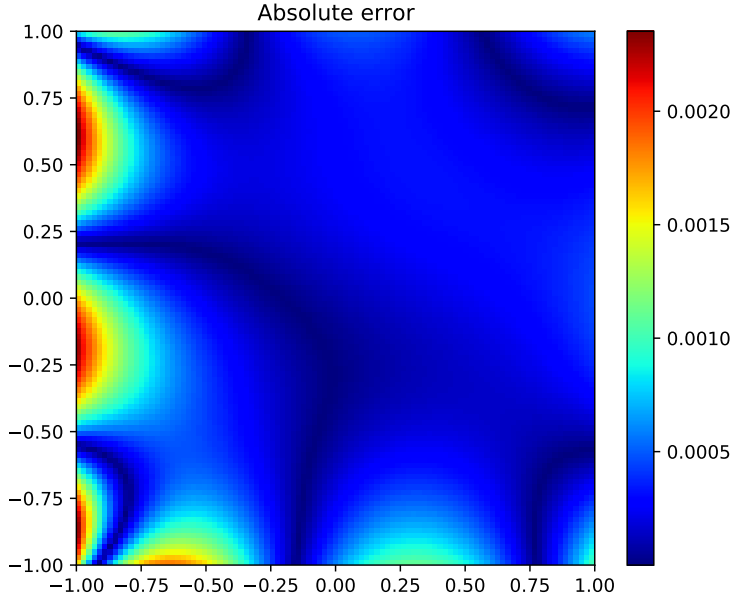
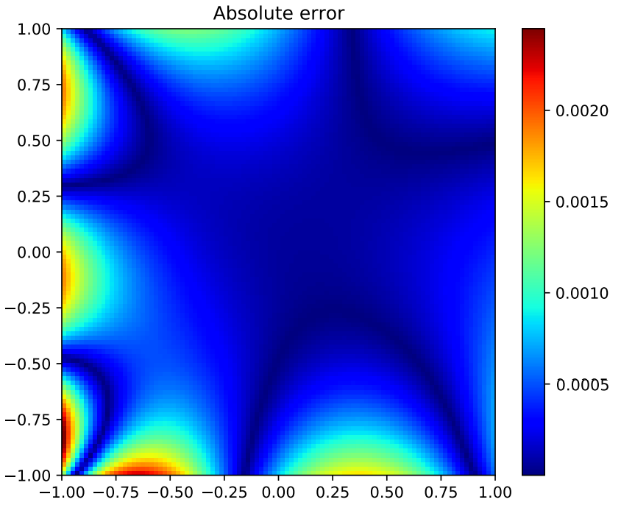
During the Adam training phase, the value of the loss terms changes more gently, and when using the L-BFGS-B optimizer, the loss term changes drastically and the model converges quickly. Notice that the PDE residual loss term is two orders of magnitude larger than the boundary loss term.

In Fig 3-2, we compare the predicted solution with the exact solution and report the point-wise absolute error between them. The point-wise absolute error of VgEmPINN and EmPINN is of order , while the error of PINN, gPINN, VPINN, and hp-VPINN is of order . We can see that the VgEmPINN method achieves the best performance.

|  |  |
| --- | --- |
|  |  |

1. Comparison of the exact solution with the predicted solution given by VgEmPINN.





1. The point-wise absolute error of VgEmPINN, EmPINN, PINN, gPINN, VPINN, hp-VPINN.

Figure 3-2. Performance of different models on Poisson’s equation.

In Table 2, we compare the VgEmPINN model against the other models. To ensure fairness in the comparison of different models, we use the same hyperparameter settings in Table 1. The experimental results show that VgEmPINN is approximately two orders of magnitude lower than those of PINN, gPINN, VPINN, and hp-VPINN, which achieves the best performance of 3.20E-05 on Poisson’s equation.

|  |  |
| --- | --- |
| Methods for solving Poisson’s equation | Relative L2 error |
| PINN | 2.28E-03 |
| gPINN | 2.16E-03 |
| VPINN | 1.76E-03 |
| hp-VPINN | 1.26E-03 |
| EmPINN | 6.70E-05 |
| VgEmPINN | 3.20E-05 |

Table 2. Comparison of the relative L2 error of different methods on Poisson’s equation.

# Burgers’ equation

The Burgers’ equation is a nonlinear PDE that models the propagation and reflection of shock waves and has applications in various areas of applied mathematics, including fluid mechanics, nonlinear acoustics, gas dynamics, and traffic flow [27]. For smaller values of the viscosity parameter, the Burgers’ equation leads to shock formation, which is known to be a difficult problem to solve by classical numerical methods. In one space dimension, Burger’s equation along with Dirichlet boundary conditions is shown as follows:

Let the approximate solution of neural network , then the loss function is defined as follows:

Here, denotes the collocation points on initial and boundary, denotes the collocation points for and , denotes the quadrature points for .

Fig 3-3 depicts the convergence of VgEmPINN for solving the Burgers’ equation.

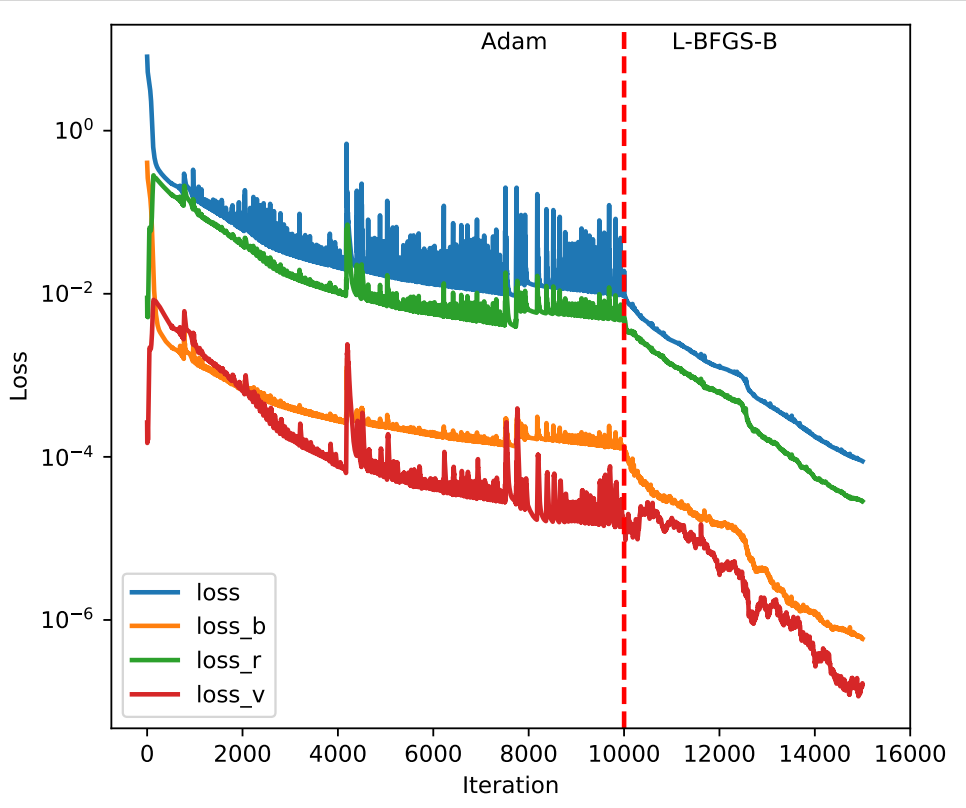
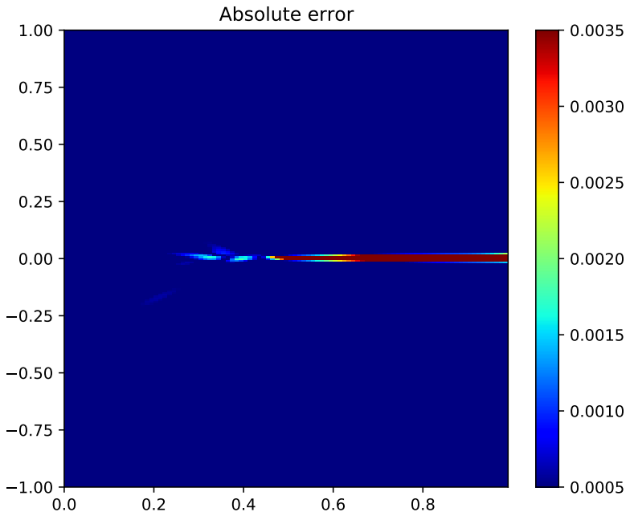
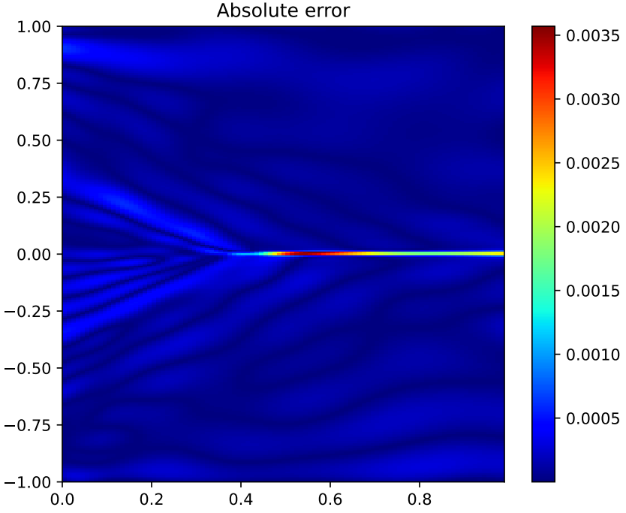


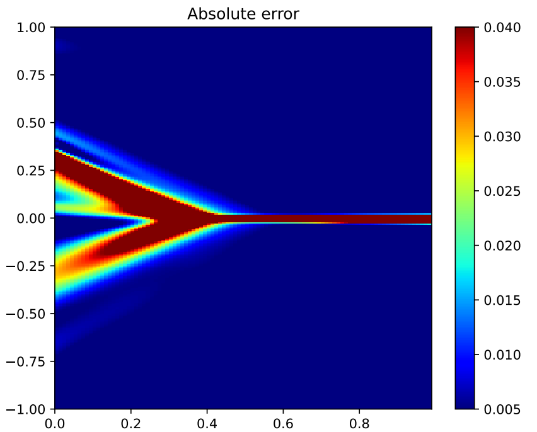
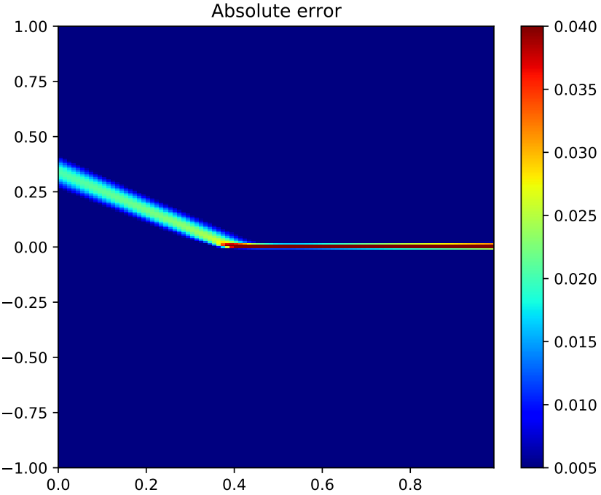
Fig 3-3 The convergence of VgEmPINN (on the log scale) on the Burgers’ equation. The Adam optimizer is used before the vertical dashed line, and the L-BFGS-B optimizer is used afterward.

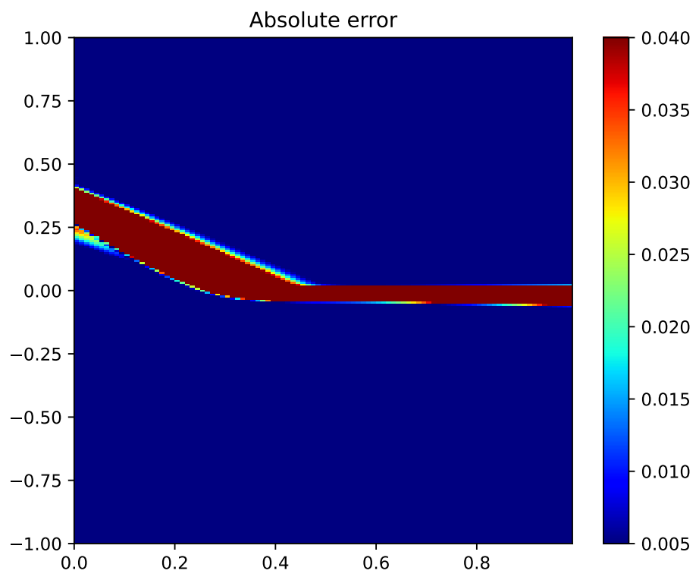
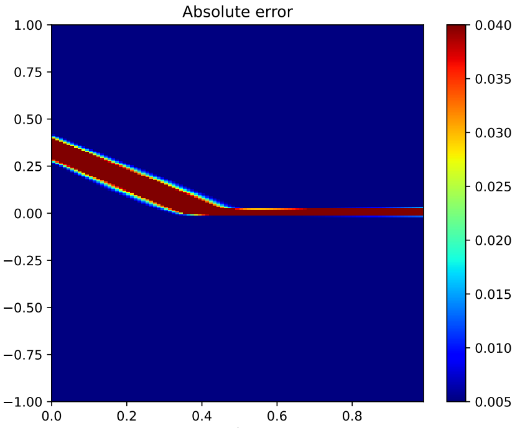
In Fig 3-4(a), we compare the predicted solution with the exact solution and report the point-wise absolute error between them.

|  |  |
| --- | --- |
|  |  |

1. Comparison of the exact solution with the predicted solution given by VgEmPINN.







1. The point-wise absolute error of VgEmPINN, EmPINN, PINN, gPINN, VPINN, hp-VPINN.

Figure 3-4. Performance of different models on the Burgers’ equation.

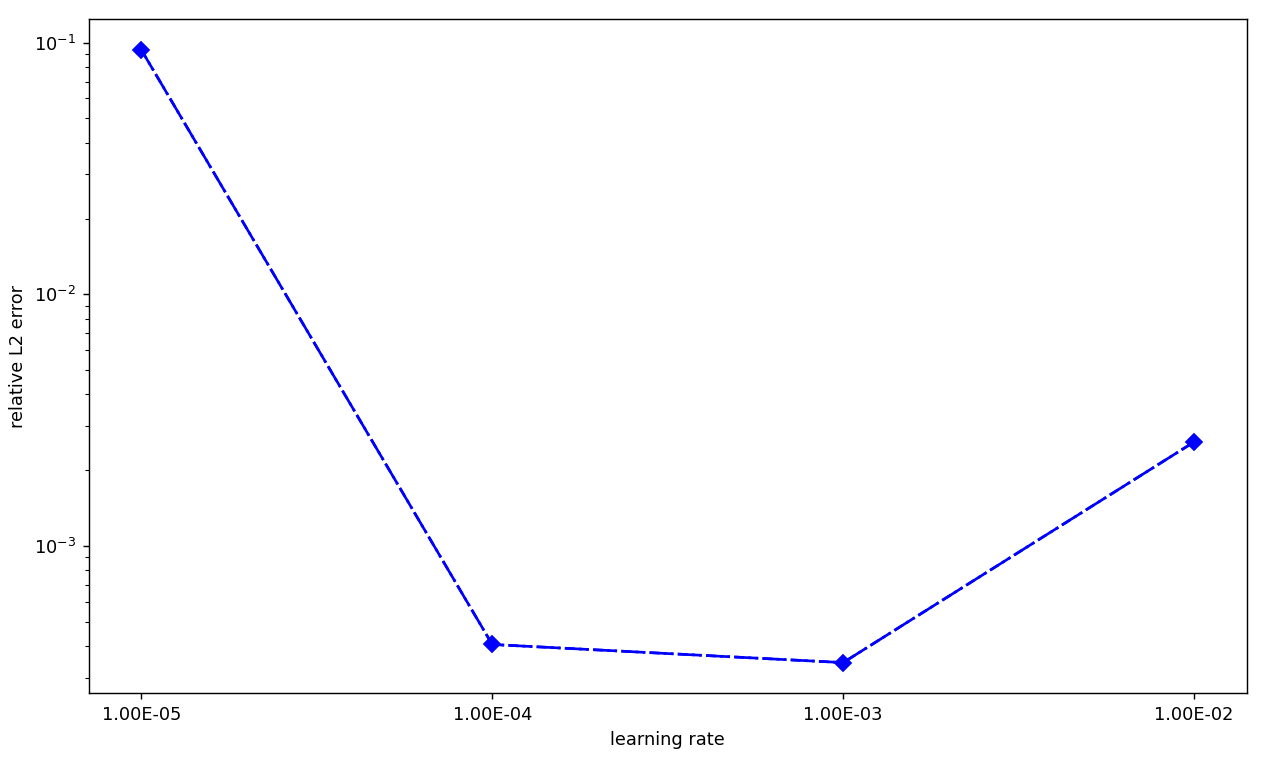
We also summarize the point-wise absolute error of different models on the Burgers’ equation in Fig 3-4(b). The point-wise absolute error of VgEmPINN and EmPINN is of order , while the error of PINN, gPINN, VPINN, and hp-VPINN is of order . As expected, VgEmPINN presents good agreement with the exact solution and achieves the smallest point-wise absolute error in the solution domain of Burgers’ problem.

Table 3. provides a detailed evaluation of the Relative L2 error of different models. It’s shown that VgEmPINN is approximately two orders of magnitude lower than those of PINN, gPINN, VPINN, and one order of magnitude lower than EmPINN, which achieves the best performance of 3.44E-04 on Burgers’ equation.

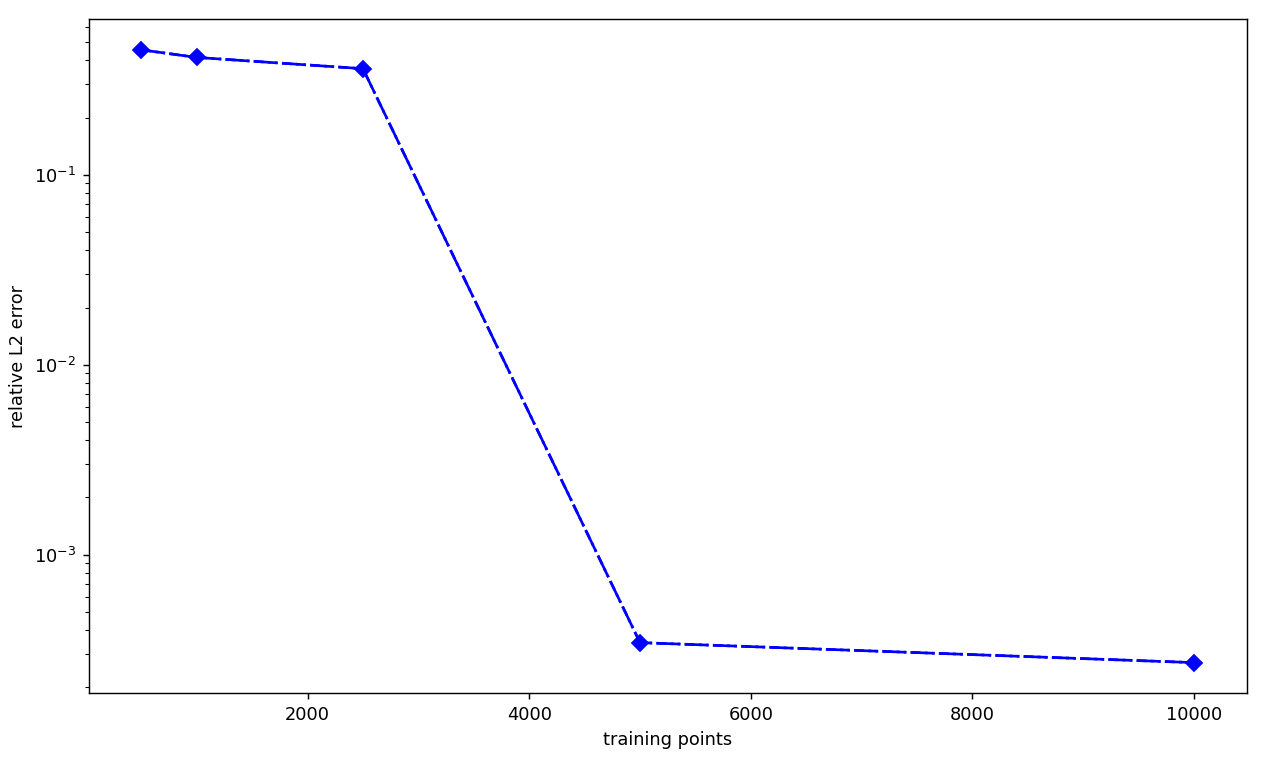
|  |  |
| --- | --- |
| Methods for solving Burgers’ equation | Relative L2 error |
| PINN | 3.50E-02 |
| gPINN | 7.43E-02 |
| VPINN | 2.77E-02 |
| hp-VPINN | 3.14E-01 |
| EmPINN | 3.78E-03 |
| VgEmPINN | 3.44E-04 |

Table 3. Comparison of the relative L2 error of different methods on the Burgers’ equation.

The tuning of hyperparameters is a fundamental component and an important process of deep learning methods. So, in the paper, we evaluate the effect of learning rate and training points. A comparison of relative L2 error for different learning rates is shown in Fig. 3-5(a). It’s shown that the range of 1.0E−03 to 1.0E−02 yields good convergence in this case. Fig. 3-5(b) depicts that when there are fewer training points, the model does not work well and is hard to converge, and when the training points are more than 5000, the model can yield good convergence.



1. relative L2 error vs. learning rate.



1. relative L2 error vs. training points.

Figure 3-5. A comparison of the relative L2 error for different learning rates and training points.

Finally, summarizing the results of two experiments on Poisson's and Burgers' equations, the applicability and better accuracy of the VgEmPINN method are verified. And we perform the study of the tuning of hyperparameters, which demonstrates that when the learning rate is in the range of 1.0E−03 to 1.0E−02 and the training points are adequate, VgEmPINN can solve the PDEs with remarkable accuracy.

# Conclusion

In this paper, we address the problem of low accuracy of the PINN method for solving PDEs by investigating aspects such as extending the dimensionality, modifying the network structure, and improving the loss function to make full use of the physical information, and we propose an improved unsupervised method for solving PDEs called VgEmPINN. Then we perform experimental validation on two equations, Poisson's equation and Burgers' equation. The results show that the VgEmPINN method can improve the accuracy of the neural network in solving PDEs and outperform the current conventional methods such as PINN by two orders of magnitude.

Solving PDEs using deep learning techniques does not require meshing of the solution domain compared to traditional numerical methods, transferring the computational time of traditional methods to the training process, which is fast when solving or simulating after the training process is completed. In addition, the proposed method can be applied as an unsupervised method, which does not require a large amount of data for training. When solving the PDE, only the form of the PDE and the corresponding initial and boundary conditions need to be known.

In future work, we will focus on studying the interpretability of the VgEmPINN method. For the method proposed in this paper, only experiments have been conducted to verify its accuracy and applicability, but the theoretical proof and interpretability studies are still lacking, which means that it’s hard to design a suitable deep neural network for the PDE system. As deep learning techniques continue to evolve algorithmically, we believe that PINN-based methods have great potential for data-efficient prediction across a wide range of physical applications.

# References

1. Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature, May 2015, vol. 521, pp: 436–444.
2. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
3. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob, Lion Jones, Aidan N Gomez, Lukasz Kaiser, and Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, 2017, pages 5998–6008.
4. Anderson, J. D. & Wendt, J. Computational Fluid Dynamics Vol. 206 (Springer, 1995).
5. Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 2019, 378:686–707.
6. A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: a survey, 2018, Journal of Machine Learning Research, vol.18, pp:1-43.
7. I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural network for solving ordinary and partial differential equations, IEEE Trans. Neural Networks, 1998, 9(5) 987-1000.
8. M. Raissi and G. E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, Journal of Computational Physics, 2018, 357: pp. 125–141.
9. M. Raissi, P. Perdikaris, and G. E. Karniadakis, Machine learning of linear differential equations using Gaussian processes, Journal of Computational Physics, 2017, 348: pp. 683–693.
10. A. D. Jagtap and G. E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, 2019, arXiv preprint arXiv:1906.01170.
11. A. D. Jagtap, K. Kawaguchi, & G. E. Karniadakis. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. Proceedings of the Royal Society A, 2020, 476(2239).
12. Ehsan Kharazmi, Zhongqiang Zhang, and George Em Karniadakis. Variational physics informed neural networks for solving partial differential equations. 2019, arXiv preprint arXiv:1912.00873.
13. E. Kharazmi, Z. Zhang, and G. Em Karniadakis, hp-VPINNs: Variational physics-informed neural networks with domain decomposition. 2020, arXiv:2003.05385.
14. J. Yu, L. Lu, X. Meng, and G. E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, 2022, Computer Methods in Applied Mechanics and Engineering, vol.393.
15. 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, Communications in Computational Physics, 2020, 28 (5):2002–2041.
16. Aditi S. Krishnapriyan, Amir Gholami, Shandian Zhe, Robert M Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. 2021, Neural Information Processing Systems (NeurIPS).
17. Li, Hao and Xu, Zheng and Taylor, Gavin and Studer, Christoph and Goldstein Tom, Visualizing the Loss Landscape of Neural Nets, NIPS 2018.
18. Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. SIAM Journal on Scientific Computing, 2021, 43(5): A3055–A3081.
19. K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks, 1989, 359–366.
20. Stein, M.: Large sample properties of simulations using Latin hypercube sampling. Technometrics 29, 143–151 (1987).
21. Chen, Xinhai, Chen, Rongliang, Wan, Qian, Xu, Rui, Liu, Jie. An improved data-free surrogate model for solving partial differential equations using deep neural networks. Scientific Reports, 2021, vol.11, pp. 1-17.
22. Sifan Wang, Hanwen Wang, Paris Perdikaris. Improved architectures and training algorithms for deep operator networks. 2021, [arXiv:2110.01654](https://arxiv.org/abs/2110.01654).
23. Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. SIAM Journal on Scientific Computing, 43(5): A3055–A3081, 2021.
24. Morales, J. & Nocedal, J. Remark on algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. ACM Trans. Math. Softw. 38, 7. https:// doi. org/ 10. 1145/ 20496 62. 20496 69 (2011).
25. Abadi, M. et al. Tensorflow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 265–283 (USENIX Association, 2016).
26. van der Walt, S., Colbert, S. C. & Varoquaux, G. The numpy array: A structure for efficient numerical computation. Comput. Sci. Eng. 13, 22–30. https:// doi. org/ 10. 1109/ MCSE. 2011. 37 (2011).
27. C. Basdevant, M. Deville, P. Haldenwang, J. Lacroix, J. Ouazzani, R. Peyret, P. Orlandi, A. Patera, Spectral and finite difference solutions of the Burgers equation, Computers & fluids 14 (1986) 23–41.