Assessing the Performance of PINN and CNN Approaches in Solving the 1D Burgers' Equation with Deep Learning Architectures

Cheng Li*

School of Computer Science and Engineering, Faculty of Innovation Engineering, Macau University of Science and Technology, Macau 999078, China

2009853vi011006@student.must.edu.mo

ABSTRACT

Deep learning has emerged as a promising approach for solving complex partial differential equations (PDEs) using data-driven methods, particularly in scenarios where traditional numerical techniques face limitations. The 1D Burgers' Equation, which combines convection and diffusion terms, is a fundamental PDE used to model various physical phenomena, including fluid dynamics and shock waves. While analytical solutions for this equation are challenging due to its nonlinearity, specific cases do have exact solutions. This research systematically evaluates the effectiveness of two popular deep learning methodologies, Physics-Informed Neural Networks (PINNs) and Convolutional Neural Networks (CNNs), in solving the 1D Burgers' Equation. A particular focus of the assessment is to explore potential bias in the training dataset. The neural networks are trained using biased samples, and their results are rigorously compared with those obtained from unbiased samples. This investigation provides insights into the performance of these deep learning architectures and their sensitivity to the quality of training data. The research reveals significant sample bias effects on neural network (NN) models. Primitive bias notably impacts Convolutional Neural Networks (CNNs) more than Physics-Informed Neural Networks (PINNs). This is attributed to differences in loss function calculation, making bias selection more straightforward for CNNs.

CCS CONCEPTS

• **Mathematics of computing** → Mathematical analysis; Differential equations.

KEYWORDS

1D Burgers' Equation, Deep Learning, PINN, CNN, Partial Differential Equation

ACM Reference Format:
Cheng Li. 2023. Assessing the Performance of PINN and CNN Approaches in Solving the 1D Burgers' Equation with Deep Learning Architectures. In *2023 4th International Conference on Machine Learning and Computer Application (ICMLCA 2023), October 27–29, 2023, Hangzhou, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3650215.3650370

*Corresponding author.

1 INTRODUCTION

Computational fluid dynamics (CFD) plays a pivotal role in simulating and understanding complex fluid behaviour, with applications spanning from aerospace engineering to environmental modelling. Traditionally, CFD relies on numerical methods to solve the Navier-Stokes equations, which describe fluid motion. While these methods have been successful, they often face challenges when dealing with turbulent flows and complex geometries, necessitating innovative approaches.

In recent years, machine learning (ML) techniques, particularly Deep Learning, have garnered substantial attention in the realm of fluid dynamics. ML models, such as neural networks, have shown promise in approximating complex functions, offering a data-driven alternative to traditional numerical methods. This paradigm shift opens up exciting possibilities for more efficient and accurate fluid dynamics simulations.

One of the fundamental PDEs frequently encountered in CFD is the 1D Burgers' Equation, which is shown as the Formula (1). This equation elegantly combines convection and diffusion terms, reflecting the nonlinear nature of fluid flow. While it admits analytical solutions for specific cases, the general form poses computational challenges, making it an ideal candidate for assessing the efficacy of ML techniques in fluid dynamics simulations.

$$\partial u/\partial t + u\partial u/\partial x = v(\partial^2 u)/(\partial x^2) \tag{1}$$

In recent years, there has been a growing interest in leveraging advanced machine learning techniques, specifically Physics-Informed Neural Networks (PINNs) and Convolutional Neural Networks (CNNs), for solving complex partial differential equations (PDEs) and addressing challenges in fluid dynamics and related fields. Physics-Informed Neural Networks (PINNs) have gained prominence for their ability to seamlessly integrate domain-specific physical laws into the learning process. These frameworks have been applied extensively in solving a wide range of nonlinear PDEs, both forward and inverse problems [1]. They have also been extended to high-dimensional surrogate modelling and uncertainty quantification in problems featuring stochastic inputs without labelled data [2]. On the other hand, Convolutional Neural Networks (CNNs) have shown promise in various applications within fluid dynamics and PDE solving. They have been proposed for solving nonlinear PDEs alongside physics-based loss functions [3]. Additionally, CNNs have demonstrated their capability to approximate solutions for complex equations in quantum systems and have been applied to problems with high-dimensional random inputs, particularly in the context of stochastic PDEs [4]. Furthermore, CNNs have been employed in fluid dynamics applications, including learning mappings from boundary conditions to velocity fields and modelling fluid flows through visual data [5]. These pioneering works have paved the way for this study, which aims to assess and compare the effectiveness of PINNs and CNNs in solving the 1D Burgers' Equation while also addressing the critical issue of sample bias in the context of machine learning for fluid dynamics applications.

The primary aim of this study is to comprehensively evaluate the performance of two prominent ML methodologies in tackling the 1D Burgers' Equation: Physics-Informed Neural Networks (PINNs) and Convolutional Neural Networks (CNNs). Beyond assessing their effectiveness, the paper delves into the crucial issue of sample bias, probing its presence and impact during the neural network training process.

Furthermore, this research signifies a significant step towards harnessing the potential of ML, particularly CNNs, for approximating solutions to the Burgers' Equation and predicting fluid dynamics. By utilizing both biased and unbiased training samples, we aim to shed light on the sensitivity of ML models to data quality, offering insights into their application in real-world fluid dynamics problems.

In essence, this work not only contributes to advancing our understanding of the role of ML in computational physics and engineering but also provides valuable guidance for future endeavours in utilizing ML techniques to address the complexities of fluid dynamics in practical applications.

2 METHODOLOGY

In this research, Biased Region was used to select the samples, PINN and CNN models were used to train the samples to solve the 1D-Burgers Equation.

2.1 Biased Samples Selection

This research uses biased region samples versus unbiased region samples and compares the two types of samples as training data. The advantage of selectively allocating biased region samples is that it strategically focuses the learning process of the neural network on specific regions of interest, thereby capturing key features and complex dynamics in the data more accurately. By focusing the training points on these target regions, the neural network can gain a deeper understanding of complex features that would be difficult to learn if the samples were uniformly distributed across the entire dataset. Figure 1 shows the example of the biased samples that used in the research.

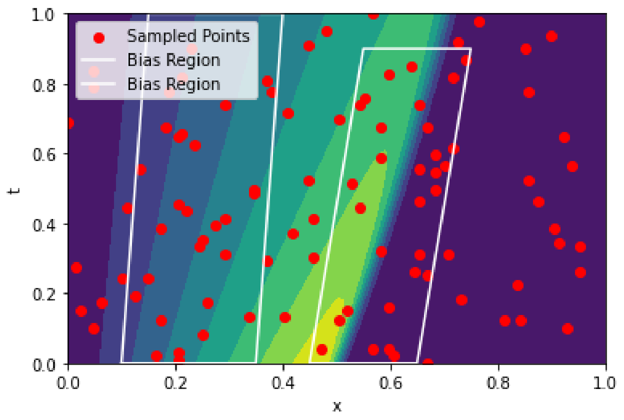


Figure 1: The example of the biased samples

2.2 Convolutional Neural Network Model

Convolutional Neural Network is a deep learning model originally designed for image processing tasks. However, over time, it has been successfully applied to a variety of fields, including natural language processing, medical image analysis, and physics modelling, such as solving the one-dimensional Burgers equation.

The core components of a CNN are the convolutional layer and the pooling layer. The convolutional layer uses convolutional kernels to perform convolutional operations on the input data to extract local features. These convolutional kernels are learnable parameters that are automatically tuned to a specific task during training [6]. In the case of solving the one-dimensional Burgers equation, the CNN can be used to recognize patterns in space and time, which helps in understanding fluid dynamics phenomena.

The pooling layer is used to reduce the size of the feature map, thereby reducing the computational load and increasing the computational efficiency of the model. This is particularly important for working with large-scale datasets and complex equations, as it reduces the time and computational resources required for training and inference.

An important feature of CNN is that it can stack multiple convolutional and pooling layers to extract features at multiple levels. This means that the network can gradually learn from low-level edge and texture features to high-level shapes and patterns [7]. When solving the one-dimensional Burgers equation, this hierarchical feature extraction can help the model capture complex phenomena in fluid flow, such as excitations and vortices.

Finally, CNNs typically include fully connected layers that are used to map features extracted from convolution and pooling layers to specific output classes. In the case of solving the one-dimensional Burgers equation, these fully connected layers can be used to predict the nature of hydrodynamic phenomena, such as the distribution of velocity or pressure fields.

In addition to solving the one-dimensional Burgers equation, CNNs play an important role in many other scientific and engineering applications. For example, they can be used to analyse medical images to help doctors diagnose diseases. In addition, CNNs are used in natural language processing tasks such as text categorization and speech recognition. In the field of autonomous driving, CNNs are used to detect and track objects on the road to ensure that vehicles are traveling safely.

CNNs are used in solving the one-dimensional Burgers equation its convolutional and pooling layers allow for efficient extraction of features, the multilayered structure allows it to handle complex data patterns, and the fully connected layer allows for the mapping of these features to the output of the actual problem. In this research, the ReLU function is chosen as the activation function and the log_cosh_losses is chosen as the loss function.

2.3 Physics Informed Neural Network Model

Physics-Informed Neural Networks (PINNs) have emerged as a groundbreaking approach in solving complex physical problems, particularly in the context of partial differential equations (PDEs). One fascinating application of PINNs lies in tackling the one-dimensional Burgers Equation, a fundamental equation in fluid dynamics and nonlinear wave phenomena at its core, a PINN is a type of neural network that integrates physics knowledge, represented as PDEs, into the network's loss function. This fusion of deep learning and physics not only enhances the network's ability to solve complex problems but also enables it to understand and respect the underlying physical principles governing the system under study.

The magic of PINNs lies in their loss function. Unlike traditional neural networks that only consider data-driven losses, PINNs go further by penalizing violations of governing equations and boundary/initial conditions. In the context of the one-dimensional Burgers Equation, this means that the network is not only learning from available data but is also constantly reminded of the PDE that governs the fluid dynamics, ensuring physical consistency [8]. To enforce the PDE and boundary conditions, automatic differentiation plays a pivotal role. During training, PINNs employ automatic differentiation techniques to compute the residuals of the PDE. These residuals quantify the extent to which the model's predictions deviate from the expected behaviour prescribed by the PDE. The network then adjusts its parameters to minimize these residuals, effectively aligning its predictions with the laws of physics.

A PINN is trained through a comprehensive optimization process that minimizes multiple loss components. These components typically include data mismatch, boundary condition deviations, and PDE residuals. By optimizing these aspects simultaneously, the network not only learns from available data but also respects the physical constraints defined by the PDE and boundary conditions [9]. This holistic training approach ensures that the network captures both observed behaviour and adheres to the underlying physics.

One of the remarkable advantages of PINNs is their efficiency in making predictions. Once trained, a PINN can rapidly predict solutions without explicitly solving the PDEs. This is a substantial departure from traditional numerical methods that require iterative solvers. The PINN leverages its learned understanding of the physics to provide quick and accurate solutions, making it a valuable tool for real-time simulations and decision-making in various applications, including fluid dynamics. In this paper, the activation function is chosen the hyperbolic tangent function (Tanh), which is a common choice in neural networks. It is defined in the OptimizedTanh class with adjustable slope and midpoint parameters.

2.4 A Generic Approach for Training Model

In the training process, crucial data including sampling points denoted as (x, t), initial points, and boundary points are provided to the model. This study used Formula (2) to calculate the exact solution for the sample points. These data serve as the foundation for guiding different components of the loss function, playing a pivotal role in training the model effectively. Through the optimization process, the model refines its parameters and adapts to the provided data, ultimately improving its performance and accuracy in solving the underlying problem or task at hand.

$$\begin{aligned} u(x,t) = x/(t+1) \div (1 + \sqrt{((t+1)}/t\_0) \\ .exp((R\_num \wedge x^2)/(4t+4))) \end{aligned} \tag{2}$$

Figure 2 shows the comparative overview of training methods, the first step is to initialize parameters and then prepare the boundary and initial conditions. After that, it I to define the loss function and train the loop. As for the loss part, the total loss of PINN includes the supervised loss and the Physics loss with certain weights, but the total loss of CNN just includes the supervised loss. After that, it is the update parameters step and check the convergence, which are before completing the training step.

3 RESULTS & DISCUSSIONS

The Initial/Final Conditions Comparison graph is used to compare the difference between the predictive effect of the model on the initial/final conditions and the true solution. In these plots, the horizontal coordinate represents the value of the spatial coordinate x, in the case of the initial time step (t = 0/final). The vertical coordinate represents the value of the solution, i.e. the predicted and true solution values of the model.

Figure 3 shows some sample results comparing the CNN model on biased and unbiased data. The research used two biased regions with 30 extra samples in each. The parameters are bias_regions = [([(0.1, 0.0), (.15, 1.0), (0.4, 1.0), (0.35, 0.0)], 30), ([(0.45, 0.0), (0.55, 0.90), (0.75,0.90),(0.65,0.0)], 30)], hidden_sizes = [6, 32, 32, 64, 32, 32], best_loss = 90.0, max_stop_iter = 90, Rnum = 500.0, num_epochs = 15000 and learning_rate = 0.001. The loss for biased samples is 0.0002 and the loss for unbiased samples is 0.0014. The biased model achieves lower error in those regions but higher overall error.

Figure 4 shows the reducing the number of hidden layers hurt performance on the biased data for the CNN model, hidden_sizes = [6, 32, 32, 64, 32]. The loss for biased samples is 0.0000 and the loss for unbiased samples is 0.0192. It shows network capacity is important for effectively learning from biased samples.

Figure 5 shows the PINN model, the biased sampling leads to lower errors in the target regions but higher overall error compared to the unbiased case. The parameters are bias_regions = [([(0.1, 0.0), (.15, 1.0), (0.5, 1.0), (0.35, 0.0)], 30), ([(0.45, 0.0), (0.55, 0.90), (0.75,0.90),(0.65,0.0)], 30)], num_epochs = 15000, learning_rate = 0.001, Rnum = 500.0, hidden_sizes = [40,40,40,40], max_stop_iter = 90, stop_iter = 0 and batch_size = 20.

Figure 6 shows add more hidden layers also didn't help much, showing that the PINN model has difficulty fitting the specific biased samples, hidden_sizes = [40,40,40,40,40].

4 CONCLUSION

In future research, we aim to conduct more rigorous investigations into bias regions, specifically assessing the impact of varying bias percentages within each region. Notably, our findings suggest

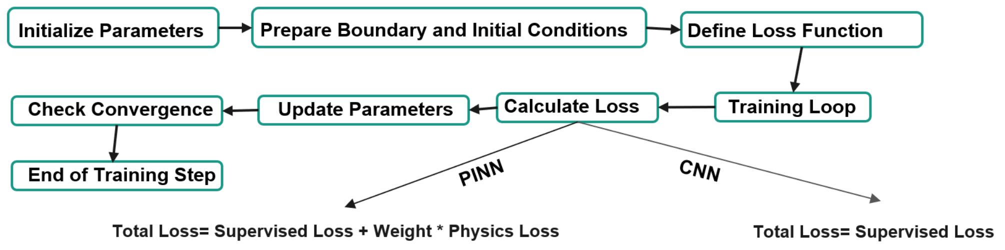
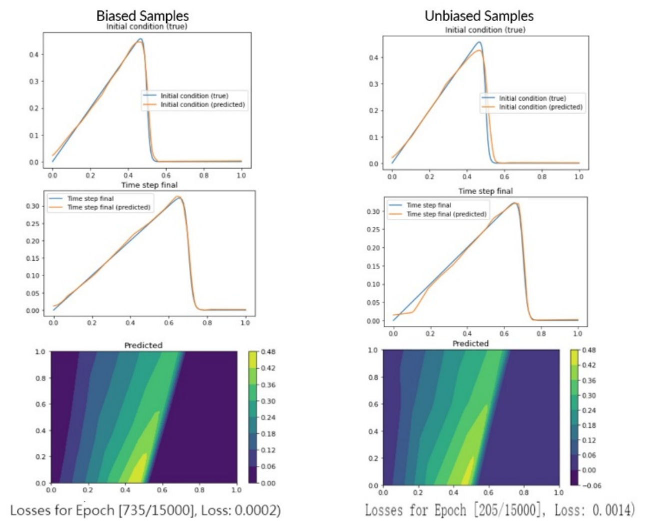
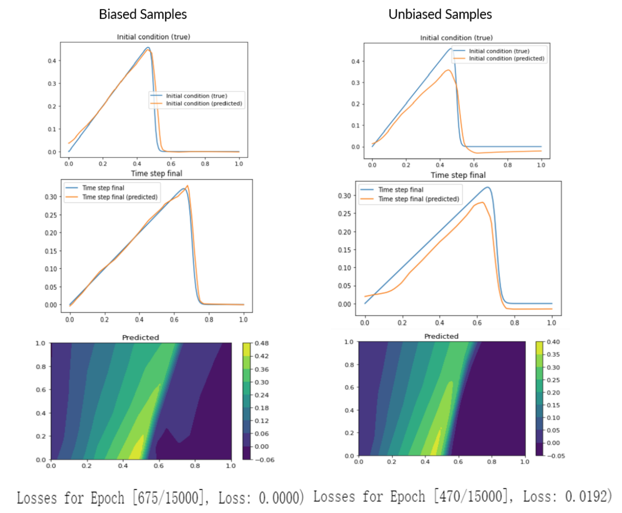


Figure 2: The comparative overview of training methods



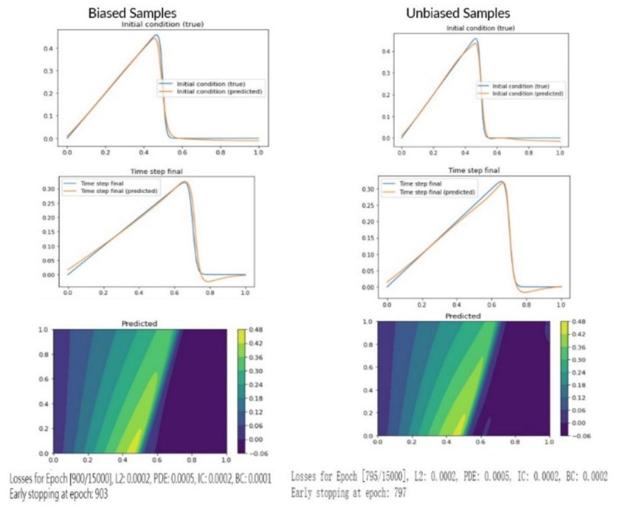
Losses for Epoch [735/15000], Loss: 0.0002)
Losses for Epoch [205/15000], Loss: 0.0014)

Figure 3: CNN sample results



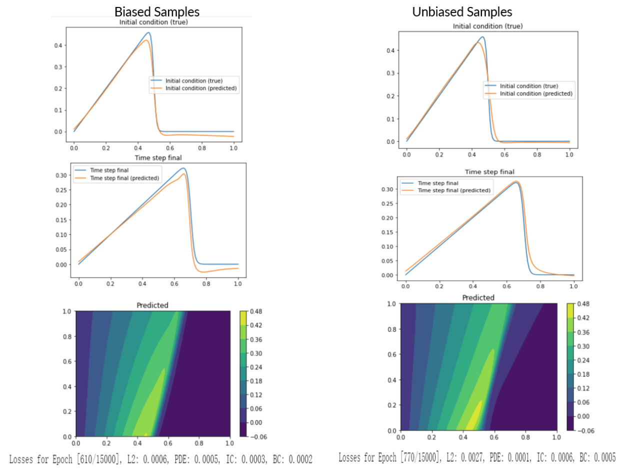
Losses for Epoch [675/15000], Loss: 0.0000)
Losses for Epoch [470/15000], Loss: 0.0192)

Figure 4: CNN sample results after changing layers



Losses for Epoch [900/15000], L2: 0.0002, PDE: 0.0005, IC: 0.0002, BC: 0.0001
Early stopping at epoch: 903
Losses for Epoch [795/15000], L2: 0.0002, PDE: 0.0005, IC: 0.0002, BC: 0.0002
Early stopping at epoch: 797

Figure 5: PINN sample results



Losses for Epoch [610/15000], L2: 0.0006, PDE: 0.0005, IC: 0.0003, BC: 0.0002
Losses for Epoch [770/15000], L2: 0.0027, PDE: 0.0001, IC: 0.0006, BC: 0.0005

Figure 6: PINN sample results after changing layers

that Convolutional Neural Networks exhibit reduced effectiveness when applied to nonlinear Partial Differential Equations (PDEs), highlighting the need for a stronger emphasis on Physics-Informed Neural Networks [10]. Our results indicate a discernible influence of sample bias on neural network models, with CNNs being more susceptible to the effects of basic sample bias than PINN models. Interestingly, it appears that selecting appropriate sample bias is a relatively simpler task for CNNs compared to PINNs, owing to the complexity of the loss function calculation in the latter. Thus, when choosing sample bias, a comprehensive evaluation of all relevant losses should be conducted, rather than solely focusing on the nonlinearity of the underlying problem.

ACKNOWLEDGMENTS

Author thanks Hsien Shang and Somdeb Bandopadhyay for their help and guidance during this research.

REFERENCES

[1] Raissi, M., Perdikaris, P., Karniadakis, G.E. 2019. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378, 686–707.
[2] Raissi, M., Perdikaris, P., Karniadakis, G.E. 2017. Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations, arXiv preprint, arXiv:1711.10561, 2017.
[3] Cai, S., Mao, Z., Wang, Z., Yin, M., Karniadakis, G.E. 2022. Physics-informed neural networks (PINNs) for fluid mechanics: a review, Acta Mech. Sin. https://doi.org/10.1007/s10409-021-01148-1.
[4] Krizhevsky, A., Sutskever, I., Hinton, G.E. 2012. ImageNet classifcation with deep convolutional neural networks. Adv Neural Inf Process Syst 25:1097–1105.
[5] Raissi, M., Karniadakis, G.E. 2017. Hidden physics models: machine learning of nonlinear partial differential equations, 2017, arXiv:1708.00588.
[6] Ambikasaran, S., Li, J.Y., Kitanidis, P.K., Darve, E. 2013. Large-scale stochastic linear inversion using hierarchical matrices. Comput. Geosci. 17 (6), 913–927. https://doi.org/10.1007/s10596-013-9364-0.
[7] Bengio, Y., Delalleau, O., Roux, N. 2006. The curse of highly variable functions for local kernel machines. Adv. Neur. In 18, 107–114.
[8] Bottou, L., Bousquet, O. 2008. The tradeoffs of large scale learning. Adv. Neur. In 20, 161–168.
[9] Broyden, C.G. 1965. A class of methods for solving nonlinear simultaneous equations. Math. Comput. 19 (92), 577–593.
[10] Hoult, D.I. 2000. The principle of reciprocity in signal strength calculations-A mathematical guide. Concepts Magn. Reson.