XLVI CILAMCE
VITÓRIA - ES - BRAZIL - 2025
XLVI IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGENEERING

Convolutional neural network techniques for the efficient solution of fluid flow in porous media

Diogo M de Almeida[1], Icaro A M de Souza[2], Darlan K. E. de Carvalho[3], Alessandro R. E. Antunes[4], Paulo R. M. Lyra[3], Ramiro B. Willmersdorf[3]

[1]*Institute for Petroleum and Energy Research, Universidade Federal de Pernambuco, Av. da Arquitetura, Recife-PE, Brasil.*
[2]*Center for Informatics, Universidade Federal de Pernambuco, Av. Jorn. Aníbal Fernandes Recife-PE, Brasil.*
[3]*Mechanical Engineering Department, Universidade Federal de Pernambuco, Av. da Arquitetura, Recife-PE, Brasil.*
[4]*Technological Center, Agreste Campus, Universidade Federal de Pernambuco, Av. Marielle Franco, Caruaru-PE, Brasil*
diogo.malmeida@ufpe.br, iams@cin.ufpe.br, Darlan.ecarvalho@ufpe.br, alessandro.antunes@ufpe.br, paulo.lyra@ufpe.br, ramiro.willmersdorf@ufpe.br.

Abstract. Neural operators, such as Physics-Informed Neural Networks (PINNs), the Fourier Neural Operator (FNO), and the Deep Operator Network (DeepONet), have emerged as promising alternatives to classical numerical methods and modern iterative solvers for addressing dynamic problems involving fluid flow. These neural architectures offer the potential for computational efficiency, improved generalization across varying physical conditions, and the ability to learn mappings directly between functional spaces, reducing the need for pointwise numerical simulations. However, they still fall short of the numerical efficiency achieved by a convolutional neural networks (CNNs) in finite-dimensional tasks. Among CNN variants, U-Net has shown strong performance in solving partial differential equations (PDEs). A key limitation is its reliance on large training datasets, often impractical in reservoir modeling due to limited data availability and proprietary constraints. This study proposes a data augmentation (DA) strategy to enhance U-Net's generalization and mapping capability. The proposed approach achieved reductions of 64.79% in mean relative error and 84.97% in standard deviation over 10 runs, relative to the U-Net baseline. Moreover, it outperformed DeepONet and FNO on a widely used 2D Darcy benchmark dataset. Inference time was also analyzed, highlighting the potential of the U-Net as a surrogate model for solving thousands of PDEs.

Keywords: Darcy flow, Neural operators, Convolutional neural networks, Data augmentation, Surrogate models.

1 Introduction

Uncertainty quantification (UQ) is vital in reservoir simulation to predict subsurface behavior, especially when geological knowledge is limited and porous medium properties exhibit significant spatial variability. According to Caers [1], UQ enables the explicit representation of uncertainties associated with variables such as permeability and porosity, allowing for the generation of multiple geological models consistent with observed data. These thousands of models contribute to more reliable predictions of pressure, saturation, and production, thereby helping to reduce uncertainty. Such analysis supports informed decision-making regarding production strategies and well placement, ultimately optimizing reservoir development under uncertain conditions.

Simulating thousands of large-scale numerical models, especially when using classical iterative solution methods such as the generalized minimal residual (GMRES) method, to assess the feasibility of drilling new wells is computationally extremely expensive. Techniques such as surrogate modeling mimic the behavior of the full simulation model by using a less computationally costly function that approximates the original response function [1]. In the literature, artificial intelligence (AI) has been effectively used as a surrogate modeling approach through feedforward neural networks (FNN) and convolutional neural networks (CNNs).

FNN performs non-linear mapping from an input to an output through a fully connected architecture, where each neuron in a layer is connected to all neurons in the previous layer. On the other hand, as noted by Ronneberger et al. [2], CNNs leverage convolutional layers along with downsampling and upsampling operations to effectively

capture the spatial structure of the data. Compared to FNNs, the CNNs are more efficient for processing images and spatially structured fields. In this way, a CNN variant known as U-Net gained significant attention among machine learning methods for solving partial differential equations (PDEs) [3–5]. As described by Bhaganagar and Chambers [3], U-Net employed a U-shaped architecture with encoder and decoder paths, integrating convolutional layers and skip connections to enhance spatial feature extraction. Nevertheless, in reservoir modelling, data acquisition is often constrained by factors such as limited availability, proprietary restrictions, and confidentiality concerns [6]. Consequently, a key limitation of U-Nets in the context of surrogate modelling is their reliance on amounts of data to effectively learn pattern mappings. Thus, we hypothesize that artificially augmenting the dataset, while preserving the underlying physical constraints, can improve the accuracy of U-Nets when used as surrogate models for solving PDEs.

The structure of the paper is as follows. Section 2 presents the background and related work. Section 3 details the experimental setup. In Section 4, we present and discuss the results, including comparisons with existing approaches from the literature. Finally, Section 5 highlights findings, limitations, and future research.

2 Background and related works

Direct methods such as LU factorization are impractical for solving large-scale PDEs due to the high memory demands associated with fill-in during the factorization of sparse matrices [7]. As an alternative, Krylov subspace iterative methods, such as GMRES, offer improved memory efficiency since they avoid the explicit storage of the matrix fill-in entries. However, in the absence of an effective preconditioner, such methods suffer from slow convergence or may even fail to converge, especially in the presence of ill-conditioned or highly heterogeneous systems. To overcome this limitation, the use of preconditioning becomes essential. Janka in [8] demonstrated that smoothed aggregation algebraic multigrid (SAAMG) is a robust and scalable preconditioner for GMRES, capable of handling complex linear systems efficiently. Its algebraic nature allows for automatic, black-box implementation with minimal user input. Results in the context of Stokes-type PDEs indicated that SAAMG outperformed block LU factorization as a preconditioner for GMRES, both in terms of convergence and computational efficiency[8].

However, even GMRES with preconditioning may become computationally infeasible in the context of uncertainty quantification, where the forward problem must be solved thousands of times. As an alternative, surrogate modeling techniques replicate the behavior of the full simulation model using computationally cheaper functions that approximate the original response function [1]. A line of research has proposed surrogate modeling by learning mesh-free, infinite-dimensional operators using neural networks, known as neural operators, such as DeepONet [9], the Fourier neural operator (FNO)[10], the wavelet neural operator (WNO)[11], and the U-Net wavelet neural operator (U-WNO)[4]. DeepONet has not yet produced efficient numerical algorithms comparable to the success of convolutional neural networks in finite-dimensional settings, primarily due to the high computational cost of its evaluation [10]. A key limitation of FNO lies in its use of FFT-based basis functions, which lack spatial localization and thus hinder performance under complex boundary conditions [11]. WNO, in turn, suffers from a loss of high-frequency information due to the finite-scale nature of wavelet decomposition [4]. Moreover, U-WNO incurs about 40% higher training cost than WNO, despite achieving improved accuracy [4].

A promising approach for solving PDEs is based solely on U-Net. Despite being neither mesh-free nor infinite-dimensional, it remains effective in terms of both accuracy and computational efficiency. As mentioned by Bhaganagar and Chambers in [3], data-driven methods such as U-Nets have gained significant attention in the field of surrogate modeling for solving PDEs. U-Net is a finite-dimensional artificial neural network architecture with a characteristic 'U'-shaped structure, composed of convolutional layers and skip connections, specifically designed to effectively capture and map spatial patterns. It was originally proposed for image segmentation tasks [2]. The results in [3] showed that an optimized stacking of U-Nets was capable of capturing the high-dimensional mathematical features of the elliptic operator, achieving faster convergence than the iterative multigrid method.

2.1 Problem formulation

This study considers the same physical problem examined in [4, 10, 11], namely the two-dimensional single-phase flow governed by Darcy's equation, where the pressure field is solved under homogeneous Dirichlet boundary conditions. The input data were obtained by constructing different permeability fields $a(x, y)$, modeled as samples from a random Gaussian field whose covariance is defined by an elliptic operator. To generate these random fields, an auxiliary problem was solved on a square domain $\Omega = [0, 1]^2$, discretized on a uniform grid of size $n_x \times n_y$ with $N = n_x \times n_y$ internal points. Each realization $\xi(x, y)$ was sampled from a zero-mean Gaussian process with covariance $C = (-\Delta + 9I)^{-2}$, where the Laplace operator Δ was defined with homogeneous Neumann

boundary conditions. This auxiliary problem in Eq. (1),

$$(-\Delta + 9I)^2 \, \xi = \eta, \quad \eta \sim \mathcal{N}(0, I), \tag{1}$$

was discretized using second-order central finite differences. The permeability field $a(x, y)$ was obtained from $\xi(x, y)$ through the mapping of Eq. (2):

$$a(x, y) = \psi(\xi(x, y)) = \begin{cases} perm_h, & \text{if } \xi(x, y) > 0, \\ perm_l, & \text{otherwise}, \end{cases} \tag{2}$$

where $perm_h$ is the high permeability value, and $perm_l$ is the low permeability value. This procedure yields a bimodal permeability distribution, as adopted in the literature to simulate heterogeneous porous media [4, 10, 11]. We focused on the steady-state single-phase incompressible flow in a heterogeneous porous medium, governed by mass conservation and Darcy's law, which states that the flux is proportional to the pressure gradient, as expressed in Eq. (3).

$$\nabla \cdot \mathbf{u} = f(x, y), \quad \text{with } \mathbf{u} = -\, a(x, y) \nabla u(x, y), \quad \text{in } \Omega, \tag{3}$$

where $u(x, y)$ is the pressure field, $u = 0$ on $\partial\Omega$, and $f(x, y) = 1$ represents a constant source term. The problem was discretized using the finite volume method with harmonic averaging to ensure local flux conservation.

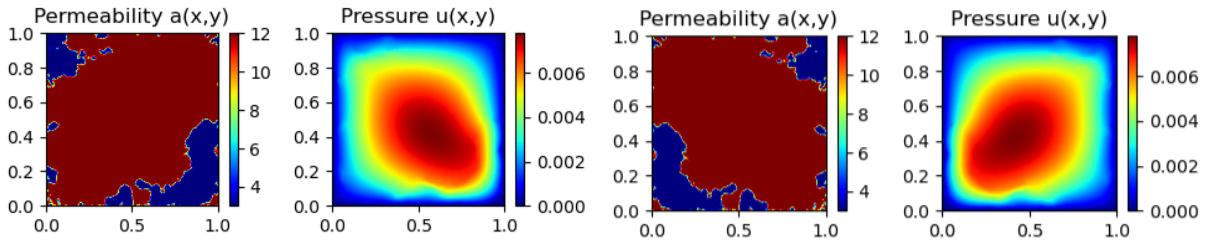
2.2 Data augmentation

Data augmentation (DA) plays a critical role in neural network training, as it increases the diversity of available data without requiring the collection of new samples [12]. Lino et al. [13] described that DA in CNN training reduces overfitting and improves generalization in fluid dynamics tasks. Regarding U-Nets, DA has proven effective for medical image segmentation. For example, Ronneberger et al. [2] used random elastic deformations, while Nemoto et al. [12] employed horizontal flipping to double the dataset size. Among artificial DA techniques, horizontal and vertical flips could enable an increase in the dataset while preserving the physical constraints of the problem. This is particularly valid for the 2D Darcy problem described in Subsection 2.1, which is analyzed spatially over a field where such flips do not interfere with the underlying physical laws. Therefore, this augmentation strategy will be employed in our experiments to validate the proposed hypothesis: artificially augmenting the dataset while preserving the underlying physical constraints can enhance the accuracy of U-Nets when used as surrogate models for solving PDEs.

3 Experimental setup

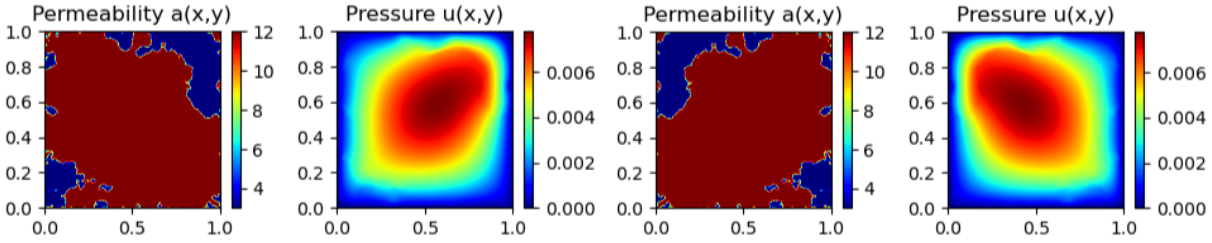
The two datasets developed in this work were generated numerically by solving the two-dimensional Darcy equation of Subsection 2.1. The domain was discretized on a uniform grid of size $n_x = 512 \times n_y = 512$, totaling $N = 512^2$ internal points. For the low-contrast dataset, the high and low permeability values are set to $perm_h = 12.0$ and $perm_l = 3.0$, respectively. For the high-contrast dataset, the values are $perm_h = 1$ and $perm_l = 10^{-4}$. The discrete linear system was solved using the GMRES method preconditioned with SAAMG, implemented via the `scipy` and `pyamg` Python libraries [14], using the following configuration: $rtol = 10^{-8}$, $restart = 300$, and $maxiter = 1000$. A total of $n_{\text{samples}} = 1024$ samples was generated for each dataset. The input $a(x, y)$ (permeability) and the corresponding solution $u(x, y)$ (pressure) are stored. These data constitute the datasets used to train the machine learning-based models. After that, the datasets were split as follows: the first 70% for training, followed by 15% for validation, and the last 15% for testing. The training set was used to train the model, the validation set to tune the hyperparameters, and the test set to evaluate the generalization performance on out-of-sample data. The DA technique applied was horizontal and vertical flipping, which made it possible to quadruple the number of samples in the training and validation sets. Figures 1 and 2 illustrate the DA process using horizontal and vertical flips. Figure 1(b) shows the horizontal flip of the original sample from Figure 1(a). Figure 2(a) presents the vertical flip of the original sample from Figure 1(a), while Figure 2(b) shows the vertical flip of the horizontally flipped sample from Figure 1(b). Finally, the samples were randomly reassigned to enhance diversity while keeping the training and validation sets distinct.

The surrogate model adopted to approximate the solution of the Darcy equation is a two-dimensional U-Net architecture. Figure 3 exemplifies a U-Net with a single block in the encoder and decoder. The network receives as input a 2D tensor of shape (n_x, n_y), representing the discretized permeability field $a(x, y)$, and returns an output tensor of the same shape, approximating the pressure field $u(x, y)$. The architecture consists of three main components: encoder, bottleneck, and decoder. The encoder progressively reduces the spatial dimensions through



(a) Original sample: $a(x, y)$ as input and $u(x, y)$ as output. (b) Flipped sample: $a(x, y)$ as input and $u(x, y)$ as output.

Figure 1. Horizontally flipped sample.



(a) Flipped sample: $a(x, y)$ as input and $u(x, y)$ as output. (b) Flipped sample: $a(x, y)$ as input and $u(x, y)$ as output.

Figure 2. Vertically flipped samples corresponding to the samples shown in Figure 1.

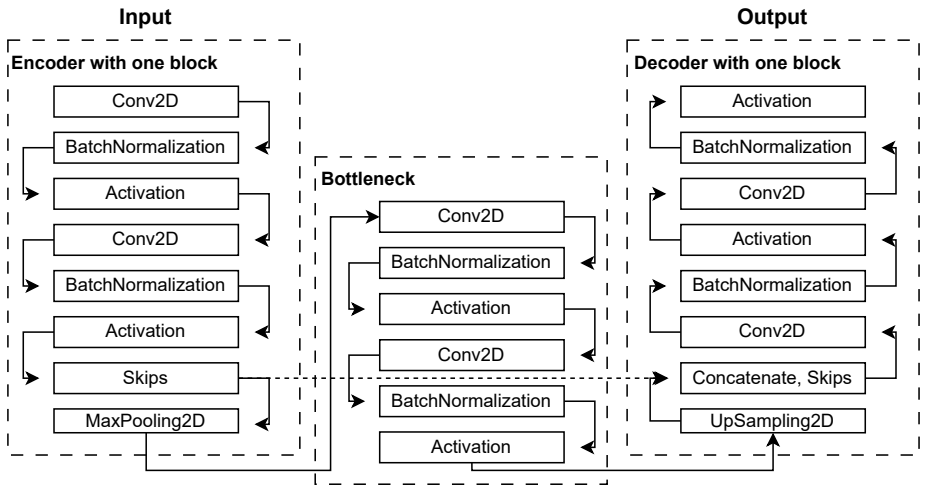


Figure 3. U-Net architecture considering a single block in the encoder and decoder.

a series of convolutional blocks followed by max-pooling operations. The number of convolutional blocks was set to 7, each block is composed of two `Conv2D` layers with a kernel size of 3, each followed by batch normalization and a nonlinear activation function. The number of filters in the i-th block is given by $F_i = F_0 \cdot 2^i$, where F_0 is the base number of filters. At the lowest resolution level, the bottleneck block consists of two `Conv2D` layers with $F_B = F_0 \cdot 2^{N_c}$ filters (where N_c is the number of convolutional blocks), also followed by batch normalization and activation functions. The decoder symmetrically increases the spatial resolution using upsampling layers (via nearest-neighbor interpolation), followed by concatenation with the corresponding encoder feature maps via skip connections. Each upsampled feature map is passed through two convolutional layers that mirror the encoder structure, using the same kernel sizes and activations. The final output layer is a `Conv2D` with the same number of filters as the input layer and a sigmoid activation, ensuring that the network's output has the same spatial dimensions and number of channels as the input. The model was trained using the Adam optimizer with a learning rate of 0.0001 for a maximum of 1000 epochs. Early stopping was applied with a patience of 20 epochs. A batch size of 8 was used, and the loss function was defined in Eq. (4) as the relative error L_{rel} between the predicted and reference solutions:

$$L_{\text{rel}} = \frac{\|\hat{u} - u\|_2}{\|u\|_2 + \varepsilon} \tag{4}$$

where $\hat{u}$ denotes the network prediction, u is the ground truth, $\|\cdot\|_2$ represents the Euclidean norm (i.e., L^2 norm), and $\varepsilon = 10^{-12}$ is a small constant added to avoid division by zero. The architecture was parameterized

by the number of filters in the range $[4, 8, 12]$ and activation functions `ReLU` or `tanh`. These hyperparameters were selected on the validation set according to the smallest relative L_{rel} error between predicted and reference solutions. The U-Net was implemented in Python [14] using TensorFlow [15] and Keras [16]. The experiments were conducted on a single core of an AMD Ryzen 9 5900XT processor with 64GB of RAM and an RTX 4070 GPU.

4 Results

Table 1 illustrates the mean relative error L_{rel} on the test set for both low- and high-contrast datasets on a 512×512 grid, comparing the performance of the U-Net model with and without DA through vertical and horizontal flips. The first row lists the methods, i.e., the standard U-Net (without DA) and U-Net$_{flip}$, which incorporates the proposed flipping-based DA technique. The subsequent rows report the mean and standard deviation of L_{rel} over 10 runs on the test set for the low-contrast dataset, followed by the results for the high-contrast dataset. The proposed method demonstrated a 35.23% decrease in the mean and a 31.49% decrease in the standard deviation of the L_{rel}, compared to the baseline U-Net, for the low-contrast dataset. In the high-contrast case, the improvements were even more pronounced, with reductions of 64.79% in the mean and 84.97% in the standard deviation.

Table 1. Mean and standard deviation of L_{rel} over 10 runs on the test set for low- and high-contrast datasets, comparing U-Net performance with and without data augmentation (vertical and horizontal flips).

	U-Net	**U-Net$_{flip}$**
Low contrast dataset	$1.93 \times 10^{-2}\,(\pm 3.97 \times 10^{-3})$	$1.25 \times 10^{-2}\,(\pm 2.72 \times 10^{-3})$
High contrast dataset	$6.76 \times 10^{-2}\,(\pm 1.55 \times 10^{-2})$	$2.38 \times 10^{-2}\,(\pm 2.33 \times 10^{-3})$

Figure 4 shows a visual comparison between the ground truth solution (left) and the U-Net$_{\text{flip}}$ prediction (right) for a sample from the low-contrast dataset. The difference obtained by U-Net$_{flip}$ is visually negligible for the low-contrast dataset. The relative error for this sample is $L_{\text{rel}} = 1.18 \times 10^{-2}$.

The same observation holds for the high-contrast dataset, as illustrated in Figure 5. The visual difference between the ground truth and the prediction from U-Net$_{flip}$ remains minimal, despite the relative error being higher compared to that obtained for the low-contrast dataset. The relative error for this sample is $L_{\text{rel}} = 2.17 \times 10^{-2}$.

Table 2 shows the time in seconds required to solve 1000 instances of the 2D Darcy problem using each method implemented in this work. The inference time of a neural network refers to the time the model takes to produce an output from a new input after it has been trained. For comparison purposes, computations were performed using a single CPU core and parallel CUDA cores on the GPU. It is observed that GMRES + SAAMG took longer to converge on the high-contrast dataset due to the ill-conditioning of the system matrix caused by the high permeability contrast, which varies from $1\text{e}{-4}$ to 1. The ill-conditioning increased the runtime of GMRES + SAAMG by more than six times (5577.41 s) compared to the low-contrast dataset (835.74 s). It is also noted that the direct LU method is the fastest on the CPU; however, it is severely limited by memory requirements as

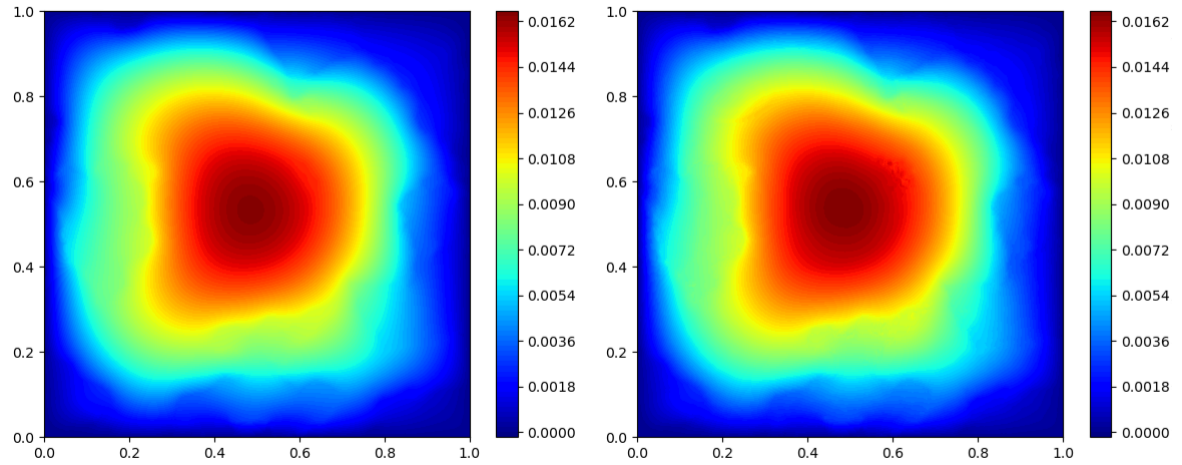


Figure 4. Low-contrast dataset. Visual comparison between the ground truth solution (left) and the U-Net$_{flip}$ prediction (right) for one sample ($L_{\text{rel}} = 1.18 \times 10^{-2}$) out of 153 on test set.

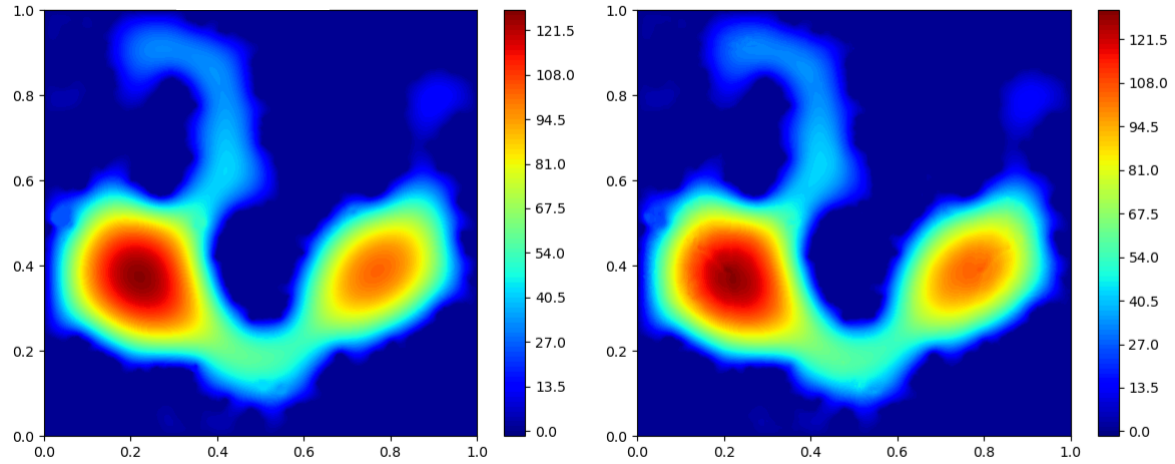


Figure 5. High-contrast dataset. Visual comparison between the ground truth solution (left) and the U-Net$_{flip}$ prediction (rigth) for one sample ($L_{\text{rel}} = 2.17 \times 10^{-2}$) out of 153 on test set.

Table 2. Inference time (in seconds) for 1000 samples on a 512×512 grid using U-Net$_{flip}$, compared to GMRES+SAAMG and the LU direct solver.

	GMRES + SAAMG (CPU)	**LU direct (CPU)**	**U-Net$_{flip}$ (CPU)**	**U-Net$_{flip}$ (GPU)**
Low contrast dataset	835.74	54.60	94.64	6.47
High contrast dataset	5577.41	55.99	96.94	6.57

the mesh size increases [7]. The most remarkable performance gain is achieved by the U-Net when executed on the GPU, reaching an inference time nearly 1000 times faster than the GMRES + SAAMG method (6.57 s vs. 5577.41 s). These results support the research hypothesis that artificially augmenting the dataset can improve the accuracy of U-Nets while maintaining fast inference, thereby making them suitable for use as surrogate models in solving thousands of PDEs.

Table 3. Comparative results are provided with those reported in [4], using the dataset available in [17], which has a resolution of 421×421 corresponding to the problem described in Section 2.1. The resolution was resized in each work due to hardware constraints and architectural requirements of the methods.

	DeepONet [4]	**WNO** [4]	**U-Net**	**FNO** [4]	**U-Net$_{flip}$**	**U-WNO** [4]
Resolution	85×85	85×85	512×512	85×85	512×512	85×85
Error L_{rel}	2.98×10^{-2}	1.84×10^{-2}	1.31×10^{-2}	1.08×10^{-2}	$\mathbf{6.61 \times 10^{-3}}$	6.20×10^{-3}

Table 3 presents a comparison between our proposed method and the results reported in [4] for a single run, across different neural operator approaches. The same dataset available in [17] was used, with a spatial resolution of 421×421. The main distinction lies in the dataset configuration: while [4] uses 1100 samples, the proposed approach employs 1024 samples. Moreover, in [4], the dataset was downsampled to a resolution of 85×85, whereas in our approach it was resized to 512×512, both due to architectural constraints of the U-Net, and to leverage the improved predictive performance observed at higher resolutions. The results demonstrate that the U-Net achieved a competitive relative error L_{rel} compared to existing methods in the literature, outperforming both DeepONet and WNO. The proposed U-Net$flip$ further surpassed FNO and achieved results comparable to those of U-WNO. However, U-WNO, as a neural operator mapping between function spaces, relies on global operations that have a high computational cost. In addition, as reported in [4], U-WNO's cost is nearly 40% higher than WNO, which constitutes a practical limitation. Therefore, despite being a finite-dimensional method, the proposed U-Net$_{flip}$ stands out due to its lower computational cost and competitive accuracy when compared to state-of-the-art neural operators.

5 Conclusions

In this work, a data augmentation technique was analyzed in the context of a U-Net architecture, aiming to enhance its ability to recognize spatial patterns for solving a Darcy-type PDE. The method achieved a reduction of 64.79% in the mean relative error and 84.97% in the standard deviation over ten runs, compared to the model without data augmentation. It also outperformed DeepONet, FNO and WNO on a widely studied benchmark dataset. Furthermore, the inference time was analyzed, confirming the feasibility of U-Net as a surrogate model for numerical solvers. The main limitation of the proposed approach lies in its finite-dimensional nature, meaning its performance is dependent on the mesh resolution used during training. As future work, it would be interesting to explore hybrid model techniques based on a divide-and-conquer strategy applied to the augmented data, with the goal of enhancing the predictive capabilities of U-Net for solving PDEs.

Acknowledgements. We would like to thank PETROBRAS (Petróleo Brasileiro S.A.) and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) for the financial support for this research.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] J. Caers. *Modeling uncertainty in the earth sciences*. John Wiley & Sons, 2011.
[2] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015.
[3] K. Bhaganagar and D. Chambers. Accelerated elliptical pde solver for computational fluid dynamics based on configurable u-net architecture: Analogy to v-cycle multigrid. *Machine Intelligence Research*, vol. 22, n. 2, pp. 324–336, 2025.
[4] W.-M. Lei and H.-B. Li. U-wno: U-net enhanced wavelet neural operator for solving parametric partial differential equations. *Computers & Mathematics with Applications*, vol. 194, pp. 272–287, 2025.
[5] M. A. Rahman, Z. E. Ross, and K. Azizzadenesheli. U-no: U-shaped neural operators. *arXiv preprint arXiv:2204.11127*, 2022.
[6] Y. Khalili and M. Ahmadi. Reservoir modeling & simulation: Advancements, challenges, and future perspectives. *Journal of Chemical and Petroleum Engineering*, vol. 57, n. 2, pp. 343–364, 2023.
[7] J. W. Pearson and J. Pestana. Preconditioners for krylov subspace methods: An overview. *GAMM-Mitteilungen*, vol. 43, n. 4, pp. e202000015, 2020.
[8] A. Janka. Smoothed aggregation multigrid for a stokes problem. *Computing and Visualization in Science*, vol. 11, n. 3, pp. 169–180, 2008.
[9] L. Lu, P. Jin, and G. E. Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.
[10] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.
[11] T. Tripura and S. Chakraborty. Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, vol. 404, pp. 115783, 2023.
[12] T. Nemoto, N. Futakami, E. Kunieda, M. Yagi, A. Takeda, T. Akiba, E. Mutu, and N. Shigematsu. Effects of sample size and data augmentation on u-net-based automatic segmentation of various organs. *Radiological Physics and Technology*, vol. 14, n. 3, pp. 318–327, 2021.
[13] M. Lino, S. Fotiadis, A. A. Bharath, and C. D. Cantwell. Multi-scale rotation-equivariant graph neural networks for unsteady eulerian fluid dynamics. *Physics of Fluids*, vol. 34, n. 8, 2022.
[14] Python Software Foundation. *Python: Version 3.12*. Available at: `https://www.python.org/`, 2024.
[15] M. Abadi and et al. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org, 2015.
[16] F. Chollet. Keras. `https://keras.io`. https://keras.io, 2015.
[17] L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, and G. E. Karniadakis. Darcy problem in a rectangular domain (piecewise constant) dataset. `https://github.com/lu-group/deeponet-fno/tree/main/data`, 2022.