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

Hybrid DeepONet Architectures for Porous Media Flow Simulation

Ezequiel S. Santos[1], Amanda C. N. de Oliveira [2], Romulo M. Silva[1], Gabriel F. Barros[1], Alvaro L.G.A. Coutinho[1]

[1]*Dept. of Civil Engineering, COPPE/Federal University of Rio de Janeiro*
ezequiel.souza@coc.ufrj.br, gabriel.barros@coc.ufrj.br, romulo.silva@coc.ufrj.br, alvaro@coc.ufrj.br
[2]*Dept. of Systems Engineering and Computer Sciences, COPPE/Federal University of Rio de Janeiro*
amandacno@cos.ufrj.br

Abstract. The solution of partial differential equations (PDEs) plays a central role in several areas of science and engineering. With the advancement of deep learning and the growing interest in data-driven methods, new approaches have emerged that aim to capture the behavior of PDE solutions. As surrogate models, Neural Operators have gained notoriety for learning parameter-to-solution operators, enabling the learning and representation of families of PDE solutions. On the other hand, Kolmogorov–Arnold Networks (KANs) represent an alternative to traditional MLP-type neural networks, replacing fixed linear connections with compositions of learnable univariate functions. This work presents hybrid models of Deep Operator Networks (DeepONet). As a distinguishing feature, we integrated different DeepONet architectures within the NVIDIA PhysicsNeMo framework, exploring both native models, such as the Fourier Neural Operator (FNO) and MLP, as well as a custom implementation of KAN. We applied these hybrid models to the context of porous media flow, such as in the Tenth SPE Comparative Solution Project, which consists of a transient two-phase flow in a 2D spatial porous medium domain. The entire modeling process, from construction to training and inference, was carried out in the PhysicsNeMo environment, with the goal of investigating the impact of combining different architectures.

Keywords: Deep Operator Networks, Fourier Neural Operators, Kolmogorov-Arnold Networks, porous media flow, Scientific Machine Learning

1 Introduction

Differential equations, whether ordinary or partial, play a fundamental role in the modeling and description of problems in numerous areas of knowledge. Their application ranges from the study of fluids in systems such as aerodynamics, hydrodynamics, and meteorology to a wide array of other disciplines. Historically, the approximation of these equations has been carried out through well-established methods, such as the Finite Element Method, the Finite Volume Method, the Boundary Element Method, the Galerkin Method, and the Finite Difference Method [1, 2]. In particular, for the modeling of flows in porous media, these methods are indispensable for simulating phenomena such as mass transport, diffusion, and a variety of other processes that govern the behavior of these systems.

Within the scope of current research, neural operators of the Deep Operator Networks (DeepONets) type, whether data-driven or physics-informed, have been established as surrogate models of great potential. Their capability lies in approximating solutions to differential equations, proving particularly advantageous in scenarios characterized by diverse geometries [3, 4]. Recent literature, moreover, has consistently validated the effectiveness of hybridization of architectures based on DeepONets [5, 6], notably those that integrate the Fourier Neural Operator (FNO) [7], a neural operator that maps function spaces through Fourier transforms for the approximation of differential equations, and the Kolmogorov-Arnold Networks (KANs) [8], which distinguish themselves from traditional perceptron neural networks (MLPs) by approximating functions through spline-type basis functions, which are learned during the training of the network.

This work investigated the impact of combining different architectures in the creation of hybrid DeepONet models for the approximation of differential equations, with a focus on flow in porous media. Utilizing the NVIDIA PhysicsNeMo framework[1], we explored hybrid architectures with native models of the framework, such as FNO and MLP, as well as a custom implementation for KANs. Our research focused on the application of these models

[1]https://github.com/NVIDIA/physicsnemo

to the Tenth SPE Comparative Solution Project [9], which simulates a two-phase transient flow in a porous medium.

2 Fluid Flow Through Porous Media Problem

In this work, we analyze the application of hybrid DeepONet models in two cases based on the SPE 10 Model 1 [9], which simulates a two-phase transient flow in a two-dimensional porous medium.

2.1 SPE 10 Model 1

For this problem, we utilize DeepONet as the base model with different architectures for simulated 2D reservoir data. DeepONet is trained to predict the total oil production, gas saturation and pressure (not shown) in the reservoir over time.

Problem Setup

In this experiment, we focus on SPE 10 Model 1[2], a small 2D gas injection problem. The model has dimensions of 2500 ft in length, 25 ft in width, and 50 ft in thickness, distributed in a grid of $100 \times 1 \times 20$. Initially, the reservoir is entirely saturated with oil. Gas is injected to displace the oil, with an injector well located on the left and a producer well on the right.

The governing equations for this problem, describing isothermal, immiscible two-phase (oil-gas) flow through a porous medium, accounting for compressibility and gravity, include:

- **Mass conservation equations** (for each phase $\alpha = o, g$):

$$\frac{\partial}{\partial t}(\phi \rho_\alpha S_\alpha) + \nabla \cdot (\rho_\alpha \mathbf{v}_\alpha) = q_\alpha \tag{1}$$

- **Darcy's law** for multiphase flow:

$$\mathbf{v}_\alpha = -\frac{k k_{r\alpha}}{\mu_\alpha}(\nabla p_\alpha - \rho_\alpha \mathbf{g}) \tag{2}$$

- **Equation of state**, relating fluid density to pressure:

$$\rho_\alpha = \rho_{\alpha,\text{ref}} \cdot \exp\left(c_\alpha (p_\alpha - p_{\text{ref}})\right) \tag{3}$$

- **Closure relation** (to close the system):

$$S_o + S_g = 1 \tag{4}$$

All computational experiments were conducted on a system utilizing an NVIDIA H100 GPU.

Dataset

The simulation was performed with the OPM Flow software[3], calculating the evolution of saturation over time. The obtained solution is validated against results from [9] and available GitHub tests[4], including comparisons of cumulative oil production (FOPT). The dataset for SPE10 Model 1 comprises simulated information including the permeability field, parameters from [9], initial reservoir conditions (fully oil-saturated), and temporal evolution of saturation data. Original OPM Flow input decks are available in the OPM data repository[5].

For SPE10 Model 1, the problem is 2D in space, with input data of shape $(100, 1, 20, N_t)$, where N_t is the number of temporal snapshots. The scalar input quantity (FGIR) is at the injection well, and the scalar output quantity (FOPR) is at the producer well. The full training data is a high-dimensional tensor of shape $(N_s, N_c, N_x, N_y, N_z, N_t)$, where N_s is the number of simulations, N_c is the number of channels, $N_x = 100$, $N_y = 1$, $N_z = 20$ are grid cell counts, and $N_t = 800$ snapshots per case. Note that N_y is suppressed as $N_y = 1$ in this model. The data from SPE10 Model 1 were validated through comparison with the results presented by Christie and Blunt [9], varying the gas injection rate (FGIR) within a range between 0.23 and 0.37, following a uniform distribution.

The DeepONet architecture is configured in two cases as follows.

1. For case 1:

[2] https://www.spe.org/web/csp/datasets/set01.htm
[3] https://opm-project.org/
[4] https://github.com/OPM/opm-tests/blob/master/spe10/SPE10.md
[5] https://github.com/OPM/opm-data/

- **Branch network inputs:** Field Gas Production Total (FGPT) and Field Gas injection rate (FGIR).
- **Trunk network inputs:** Number of days.
- **DeepONet:** combines branch and trunk to approximate Field Oil Production Total (FOPT).

The model was trained on a dataset consisting of 80 samples, with 20 samples used for testing. The data were preprocessed using Unit Gaussian Normalization.

2. For case 2:
   - **Branch network inputs:** x-coordinate, z-coordinate, permeability in the x-direction, and gas injection rates (FGIR).
   - **Trunk network inputs:** Time steps N_t.
   - **DeepONet:** combines branch and trunk to approximate Gas Saturation (SGAS).

   The model was trained on a dataset consisting of 80 samples, with 20 samples used for testing. The data were preprocessed using Unit Gaussian Normalization. The full dataset utilized for training this version of the models is approximately 5.97 GB in size.

Model Architectures

DeepONet was employed with different branch and trunk architectures to evaluate their impact on model performance and capacity. These architectural variations aimed to balance model complexity and generalization. The variations listed below show the combinations of different hybrid DeepONet models:

- **DeepONet (FNO + KAN)** uses FNO in the branch net and KAN in the trunk net.
- **DeepONet (FNO + MLP)** uses FNO in the branch net and MLP in the trunk net.
- **DeepONet (KAN)** uses KAN in both the branch and trunk nets.
- **Vanilla DeepONet** uses MLP in both the branch and trunk nets.

Case 1 In this case 1, different DeepONet configurations with varying architectures in both the branch and trunk networks were evaluated, as detailed in Table 1. Each setup was carefully optimized to balance accuracy and computational efficiency, ensuring effective training and inference.

Table 1. Detailed architectures of the DeepONet models for Case 1.

Architecture	Branch	Trunk
DeepONet (FNO + KAN)	FNO: [in_channels=1; lifting=32; (fno_layers = 2, modes = 2); mlp = [32]; out_channels = 1]	KAN: [in_features=1, 32, out_features = 1]; sp_order = 3
DeepONet (FNO + MLP)	FNO: [in_channels=1; lifting=32; (fno_layers = 2, modes = 2); mlp = [32]; out_channels = 1]	MLP: [in_features=1, 32, out_features = 1]
DeepONet (KAN)	KAN: [in_features=1, 32, out_features=1]; sp_order = 3	KAN: [in_features=1, 32, out_features=1]; sp_order = 3
Vanilla DeepONet	MLP: [in_features=1, 32, out_features=1]	MLP: [in_features=1, 32, out_features=1]

Training hyperparameters for case 1 of these DeepONet models are listed in Table 2.

Table 2. Hyperparameters used in DeepONet training for Case 1.

Parameter	Configuration
Optimizer	AdamW
Learning Rate Decay	Cosine
Decay Rate	0.01
Cosine Decay	300
Total Epochs	700

Case 2 This case 2 evaluated various DeepONet branch and trunk architectures (Table 3), optimized to balance accuracy and efficiency.

Table 3. Detailed architectures of the DeepONet models Case 2.

Architecture	Branch	Trunk
DeepONet (FNO + KAN)	FNO: [in_channels=4; lifting=16; (fno_layers = 2, modes = 4); mlp = [16,16]; out_channels = 1]	KAN: [in_features=80, 16; out_features = 20]; sp_order = 3
DeepONet (FNO + MLP)	FNO: [in_channels=4; lifting=16; (fno_layers = 2, modes = 4); mlp = [16,16]; out_channels = 1]	MLP: [in_features=80, 16; out_features = 20]
DeepONet (KAN)	KAN: [in_features=4, 16, 16, 16, 16; out_features=1]; sp_order = 3	KAN: [in_features=80, 16; out_features=20]; sp_order = 3
Vanilla DeepONet	MLP: [in_features=4, 16, 16, 16, 16; out_features=1]	MLP: [in_features=80, 16; out_features=20]

Training hyperparameters for case 2 of these DeepONet models are listed in Table 4.

Table 4. Hyperparameters used in DeepONet training for Case 2.

Parameter	Configuration
Optimizer	AdamW
Learning Rate Decay	Cosine
Decay Rate	0.01
Cosine Decay	2000
Total Epochs	3000

3 Results

This section presents and discusses the results obtained from the evaluation of various DeepONet architectures applied to transient two-phase flow in porous media. Two test cases were designed based on SPE10 Model 1: Case 1 focused on predicting Field Oil Production Total (FOPT) from injection parameters over time, while Case 2 aimed at estimating the gas saturation (SGAS) in the reservoir using spatial, permeability, and injection information. For both cases, different combinations of architectures were tested for the branch and trunk networks, including FNO, MLP, and KAN, as described previously.

The analysis includes the evaluation of each architecture based on three error metrics: Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE), computed over the test dataset. We also present qualitative comparisons between predicted and simulated data, enabling the assessment of each model's capacity to generalize to unseen inputs.

In Case 1, the focus is on the prediction of cumulative oil production (FOPT) over time, given different gas injection scenarios. The objective is to evaluate how well each DeepONet architecture generalizes from the injection parameters to the temporal evolution of FOPT. In this setting, the branch network receives gas-related inputs (FGIR and FGPT), while the trunk network is fed with temporal coordinates. Table 5 summarizes the validation performance for all DeepONet architectures evaluated in Case 1. All hybrid architectures achieved similar performance, with errors on the order of 10^{-3} for MAE and below 10^{-2} for RMSE.

Table 5. Validation of DeepONet across all hybridizations for Case 1.

Architecture	RMSE	MAE
DeepONet: considering all hybridizations	$\sim 1 \times 10^{-2}$	$\sim 7 \times 10^{-3}$

Figure 1 shows the predicted versus simulated FOPT values for two randomly selected test samples, using

the DeepONet (FNO+KAN) architecture. The hybrid model was able to reproduce the dynamics of oil production. The temporal behavior, especially during the early and late phases of production, was accurately captured. This confirms the model's capability to learn both the injection-output relationship and the underlying nonlinear temporal trends.

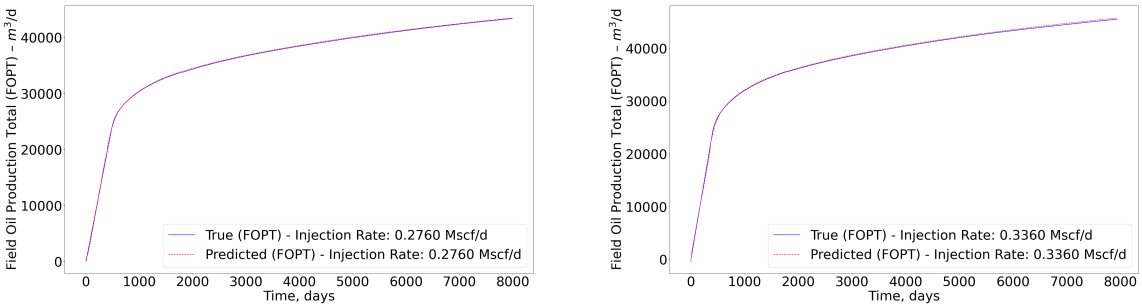


Figure 1. Validation of the hybrid DeepONet model predicting FOPT, compared to simulation results for Case 1.

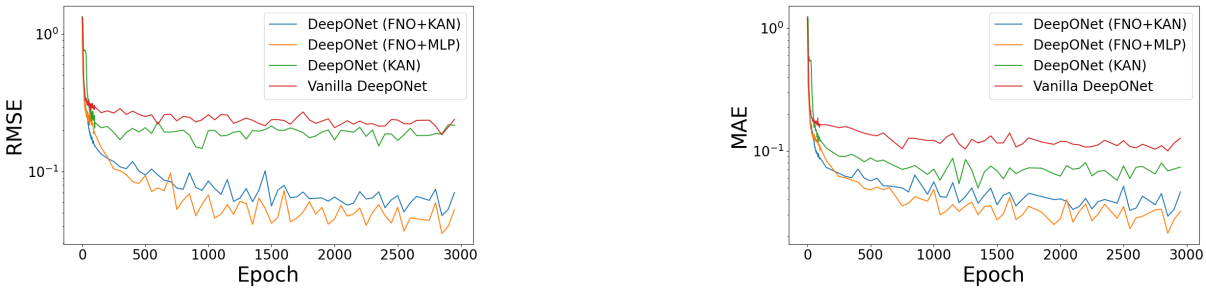
In Case 2, we evaluated the performance of DeepONet models in predicting gas saturation (SGAS) fields in a 2D reservoir over 8,000 days, under a fixed injection rate of 0.3070 Mscf/d. Inputs to the branch network included spatial coordinates and permeability values, while the trunk received temporal coordinates. Table 6 details the quantitative performance of the models.

Table 6. Validation of DeepONet across all hybridizations for Case 2.

Architecture	**RMSE**	**MAE**
DeepONet (FNO+KAN)	$\sim 5.7 \times 10^{-2}$	$\sim 3.5 \times 10^{-2}$
DeepONet (FNO+MLP)	$\sim 4.3 \times 10^{-2}$	$\sim 2.6 \times 10^{-2}$
DeepONet (KAN)	$\sim 1.3 \times 10^{-1}$	$\sim 6.2 \times 10^{-2}$
Vanilla DeepONet	$\sim 1.8 \times 10^{-1}$	$\sim 1.1 \times 10^{-1}$

The best performance was achieved by the DeepONet (FNO+MLP) model, highlighting the importance of Fourier-based feature extraction in the branch network for effectively capturing the spatial patterns inherent to this type of problem. The DeepONet (FNO+KAN) also performed well, with results only slightly inferior to those of the best model, indicating that the combination of FNO with KAN represents a alternative. The models without FNO in the branch, such as the pure KAN or MLP, presented significantly higher error rates, showing difficulty in learning from the high-dimensional input space. However, it is worth noting that the KAN-only model outperformed the Vanilla DeepONet, suggesting that the use of splines in KAN provides better representational capacity than traditional MLPs in this context.

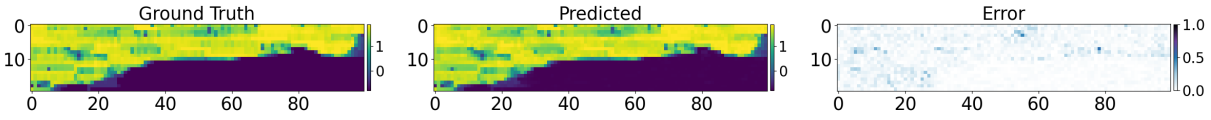
The Figures 2a and 2b present the training loss curves. The FNO-based models converged faster and to lower error levels across all metrics. In contrast, the Vanilla DeepONet and KAN-only models showed inferior convergence behavior, indicating challenges in model generalization and learning stability. As mentioned earlier, we can also see that the hybridization with pure KAN performs better than the Vanilla DeepONet.



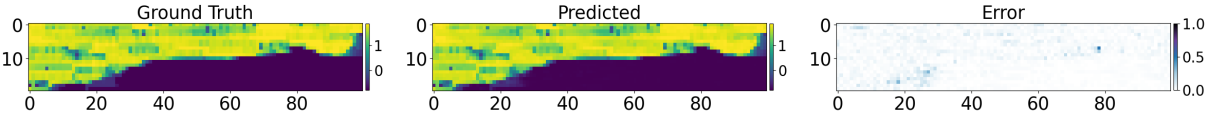
(a) RMSE. (b) MAE.

Figure 2. Training loss for different DeepONet models in Case 2.

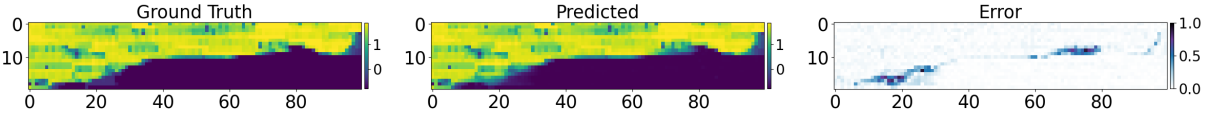
Figures 3a to 3d show the predicted and simulated SGAS fields over time for one random test sample. The DeepONet (FNO+MLP) model produced saturation profiles that closely matched the reference data, accurately capturing saturation fronts and temporal evolution. The DeepONet (FNO+KAN) also showed competitive performance, though with slightly more smoothing in sharp saturation gradients. On the other hand, models lacking the FNO as a sub-network failed to represent fine-scale spatial variations, leading to visible deviations from the simulated patterns. Despite this limitation, the hybrid model composed exclusively of KAN outperformed the Vanilla DeepONet, indicating that KAN-based architectures still offer advantages over conventional MLPs when modeling nonlinear multiphase flow dynamics in porous media.



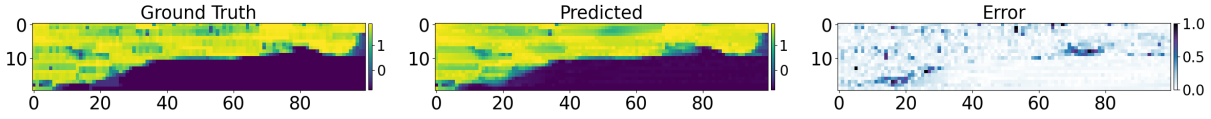
(a) DeepONet (FNO+KAN).



(b) DeepONet (FNO+MLP).



(c) DeepONet (KAN).



(d) Vanilla DeepONet.

Figure 3. Validation of DeepONet models in gas saturation (SGAS) prediction over 8,000 days, at an injection rate of 0.3070 Mscf/d. Predictions are compared against simulation results for Case 2.

4 Conclusions

This work investigated the impact of combining different architectures in the creation of hybrid DeepONet models for the approximation of differential equations, specifically focusing on flow in porous media. The research utilized the NVIDIA PhysicsNeMo framework, exploring hybrid architectures with native models from the framework, such as FNO (Fourier Neural Operator) and MLP, in addition to a custom implementation for KANs. These models were applied in the context of the Tenth SPE Comparative Solution Project, which simulates transient two-phase flow in a 2D porous media domain. Two approaches were analyzed, both based on SPE 10 Model 1:

- **Case 1:** Focused on predicting the Field Oil Production Total (FOPT) over time under different gas injection scenarios. The results showed that all hybrid DeepONet architectures achieved similar performance, with errors on the order of 10^{-3} for MAE and below 10^{-2} for RMSE. The hybrid DeepONet model (FNO+KAN), like all others, was able to accurately reproduce the oil production dynamics, capturing the temporal behavior in both the early and late production phases.

- **Case 2:** Aimed to estimate gas saturation (SGAS) in the reservoir using spatial, permeability, and injection information. In this case, the DeepONet model (FNO+MLP) achieved the best performance, highlighting the importance of Fourier-based feature extraction for effectively capturing the spatial patterns of the problem. The DeepONet (FNO+KAN) also performed well, slightly below the best model, indicating the viability of combining FNO with KAN. Models without FNO (pure KAN or MLP) showed significantly higher error rates, demonstrating difficulty in learning from the high-dimensional input space. However, the KAN-only model outperformed the Vanilla DeepONet, suggesting that the use of splines in KAN provides better rep-

resentational capacity than traditional MLPs in this context. Moreover, the FNO-based models converged faster and to lower error levels, while the Vanilla DeepONet and KAN-only models exhibited inferior convergence behavior. Visually, the DeepONet (FNO+MLP) and DeepONet (FNO+KAN) produced saturation profiles that closely matched the reference data, accurately capturing the saturation fronts and temporal evolution, whereas models without FNO failed to represent small-scale spatial variations. Nonetheless, the hybrid model composed exclusively of KAN outperformed the Vanilla DeepONet.

Hybrid DeepONet models, especially those incorporating FNO, have proven to be promising for simulating flow in porous media, with the ability to capture complex dynamics and generalize to unseen data. The integration of KAN has also proven to be a viable alternative, especially when combined with FNO, and superior to traditional MLPs in certain scenarios, highlighting the potential of these architectures for learning nonlinear operators in problems involving porous media.

Acknowledgements. This study was financed in part by CAPES, Brazil Finance Code 001. This work is also partially supported by FAPERJ, and CNPq.

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] A. Iserles. *A first course in the numerical analysis of differential equations*. Number 44. Cambridge university press, 2009.

[2] K. W. Morton and T. Sonar. Finite volume methods for hyperbolic conservation laws. *Acta Numerica*, vol. 16, pp. 155–238, 2007.

[3] 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.

[4] S. Goswami, A. Bora, Y. Yu, and G. E. Karniadakis. Physics-informed deep neural operator networks. In *Machine learning in modeling and simulation: methods and applications*, pp. 219–254. Springer, 2023.

[5] L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, and G. E. Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, vol. 393, pp. 114778, 2022.

[6] D. W. Abueidda, P. Pantidis, and M. E. Mobasher. Deepokan: Deep operator network based on kolmogorov arnold networks for mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, vol. 436, pp. 117699, 2025.

[7] 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.

[8] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

[9] M. A. Christie and M. J. Blunt. Tenth spe comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Evaluation & Engineering*, vol. 4, n. 04, pp. 308–317, 2001.