InterPore2024
Invited Student Paper Award

# ANCHORED PHYSICS-INFORMED NEURAL NETWORK FOR TWO-PHASE FLOW SIMULATION IN HETEROGENEOUS POROUS MEDIA

Jingqi Lin[1,2], Xia Yan[1,2], Kai Zhang[1,4], Zhao Zhang[3], Jun Yao[1,2]

*[1]School of Petroleum Engineering, China University of Petroleum (East China), Qingdao, China; [2]National Key Laboratory of Deep Oil and Gas, China University of Petroleum (East China), Qingdao, China; [3]Research Centre for Mathematics and Interdisciplinary Sciences, Shandong University, Qingdao, China; [4]Civil Engineering School, Qingdao University of Technology, Qingdao, China*

**Correspondence to:**
Xia Yan at jsyanxia1989@163.com,
Kai Zhang at zhangkai@upc.edu.cn

**How to Cite:**
Lin, J., Yan, X., Zhang, K., Zhang, Z., & Yao, J. (2025). Anchored Physics-Informed Neural Network for Two-Phase Flow Simulation in Heterogeneous Porous Media. *InterPore Journal*, 2(3), IPJ250825-5. https://doi.org/10.69631/ipj.v2i3nr67

**RECEIVED:** 1 Dec. 2024
**ACCEPTED:** 9 May 2025
**PUBLISHED:** 25 Aug. 2025



## ABSTRACT

In this study, we propose a tensorization-anchoring strategy based on adaptive architectures to accelerate the computation of Enriched Physics-Informed Neural Networks (EPINN) for two-phase flow simulations. Specifically, we design an adaptive tensorization mechanism for the adjacency matrix embedding, the activation function, and the skip-gated connection in EPINN, which collectively expand the neural network's (NN) parameter space for learning more generalized patterns. Moreover, we developed an anchoring strategy by establishing Anchors-EPINN (An-EPINN). By detaching tensorization parameters from the computational graph and anchoring weighted nodes to fixed positions, the NN can benefit from tensorized fusion effects while reducing high-dimensional matrix calculations during forward and backward propagation, thereby enhancing simulation efficiency. This approach reduces execution time by 31.47% in homogeneous cases and 27.91% in heterogeneous cases, while maintaining higher computational accuracy.

## KEYWORDS

Artificial intelligence for partial differential equations, AI4PDE, Simulation, Two-phase flow, Heterogeneous, Tensorizing, Adaptive architecture, Neural networks



## 1. INTRODUCTION

Recently, artificial intelligence for partial differential equations (AI4PDEs) (4) has emerged as a promising research area in machine learning. The next generation of neural networks (NNs) incorporates physical knowledge through improvements in network architecture (22), loss functions (11), and training strategies (23). Physics-informed neural networks (PINNs) (28), which are based on PDEs, learn directly

from physical laws and have demonstrated superior performance across a variety of physical problems (10). Methods similar to PINNs utilize automatic differentiation (3) to construct temporal and spatial derivatives of PDEs, enabling unsupervised learning. Additionally, methods similar to PINNs have been widely applied in fluid dynamics analysis, including the simulation of complex flow phenomena, with computational performance enhancements achieved through the design of model architectures, optimization of convergence rates, and development of specialized computational modules (40). However, these models often exhibit slow convergence speed, particularly in heterogeneous models. This is primarily attributed to significant variations in the physical parameters within heterogeneous porous media, which cause traditional PINNs to become easily trapped in complex loss landscapes under rigorous physical constraints (39). Some studies have attempted to address the convergence challenges in PINNs training through adaptive sampling strategies (33), modified loss functions (1), and decomposition methods (37). In general, the difficulty in convergence is a persistent challenge encountered in physics-constrained learning.

The latest research trend in the field of computational fluid dynamics (CFD) involves a deeper integration of physics-driven learning with traditional simulation methods (AI4CFD) (5). Numerous studies (6, 29, 31) have focused on discretizing physical constraints, proving these methods to be more suitable for discrete system simulation compared to vanilla PINNs. In a previous study (39), we used the Finite Volume Method (FVM) (13) to discretize the governing equations for two-phase flow simulation in porous media, ensuring strict mass conservation between adjacent grid cells and incorporating a well model (27) to address significant pressure gradients at sources and sinks. We also implemented hard constraints to directly encode boundary conditions into the loss function. In a separate study (35), we introduced a novel Enriched Physics-Informed Neural Network (EPINN), which embeds adjacency locations within the NN to better perceive and learn the connectivity relations. Notably, we incorporated adaptive embedding, adaptive activation functions, and adaptive gated connections to accelerate training. Increasing the parameter count of neural networks usually enhances model fitting capabilities, especially for more complex issues, including depth (21) or width (7) of the NNs, or integrating tensorized structures to increase flexibility (2, 9, 16, 18). However, this also leads to increased computational resource consumption during large-scale matrix operations. Particularly, the EPINN incorporates embedding of spatial adjacency relationships, which causes the matrix dimensions processed within the NN to correlate with the numerical model's node count. Therefore, it is imperative to investigate suitable methods to reduce computational resource consumption.

In this study, we propose a novel learning strategy to enhance the computational efficiency of EPINNs for two-phase flow simulations in heterogeneous porous media. Specifically, different hidden nodes are tensorized and adaptively weighted, while a progressive tensorization-anchoring strategy is designed. During the early training stages, the network's adjacency information, activation functions, and gated connections undergo tensorization, prompting enhanced expressive capacity. When the model converges, the adaptive parameters are detached from the computational graph and anchored to fixed hidden nodes. The strategy's effectiveness was validated through homogeneous and heterogeneous cases, demonstrating superior computational efficiency while achieving marginally higher computational accuracy.

This paper is organized as follows: First, we introduce the architecture and strategies for discrete PDE learning. Second, we describe the "tensorization-anchoring" strategy used in An-PINN. Third, we demonstrate the performance of An-EPINN across various test cases. Finally, we summarize the key contributions of this study.

## 2. METHODOLOGY

### 2.1. Physics-driven EPINN of Two-Phase Flow Simulation in Porous Media

In this study, we consider the discretized PDE based on the Darcy' law (24). The two-phase flow governing equation incorporating source and sink terms is (**Eq. 1**):

$$\phi \frac{\partial S_\beta}{\partial t} + S_\beta \phi (C_\beta + C_\mathrm{r}) \frac{\partial p}{\partial t} = \nabla \cdot (\lambda_\beta k \nabla \Phi_\beta) + q_\beta \tag{1}$$

where $\phi$ is porosity; $S$ is saturation; $t$ is time; $C$ is the compressibility coefficient; $p$ is pressure; $\lambda$ is mobility, defined as $\lambda = k/\mu$, where $k$ is permeability and $\mu$ is viscosity; $\Phi$ represents the flow potential, given by $\Phi = p - \rho g h$, $g$ is gravitational acceleration, $h$ is depth; the subscript $\beta$ is the fluid phase, which can be the wetting phase w or the non-wetting phase n; the subscript r is the porous medium. The Implicit-Pressure Explicit-Saturation (IMPES) scheme (8) is employed, **Equation 1** is derived as **Equation 2**:

$$\phi (S_\mathrm{n} C_\mathrm{n} + S_\mathrm{w} C_\mathrm{w} + C_\mathrm{r}) \frac{\partial p}{\partial t} = \nabla \cdot (\lambda_\mathrm{n} k \nabla \Phi_\mathrm{n}) + \nabla \cdot (\lambda_\mathrm{w} k \nabla \Phi_\mathrm{w}) + q_\mathrm{n} + q_\mathrm{w} \tag{2}$$

where $q_\mathrm{n}$ is velocity of the non-wetting phase; $q_\mathrm{w}$ is velocity of the wetting phase; the velocity of sources and sinks are defined as **Equation 3**:

$$q_\beta = \lambda_\beta \frac{2\pi k h}{\ln (r_\mathrm{e}/r_\mathrm{w}) + S_\mathrm{k}} (p - p_\mathrm{wf}) \tag{3}$$

where $r_\mathrm{e}$ is equivalent radius, defined as $r_e = 0.14\sqrt{\mathrm{d}x^2 + \mathrm{d}y^2}$, $\mathrm{d}x$ is the grid size in the $x$ direction, dy is the grid size in the $y$ direction; $r_\mathrm{w}$ is wellbore radius; $S_\mathrm{k}$ is skin factor; $p_\mathrm{wf}$ is the bottom hole pressure (BHP). Within the framework of the FVM, the divergence theorem is applied, the discretized form of **Equation 2** is derived as **Equation 4**:

$$R(p_i^t, p_i^{t+1}) = \frac{\phi (S_\mathrm{n} C_\mathrm{n} + S_\mathrm{w} C_\mathrm{w} + C_\mathrm{r}) V_i}{\Delta t} (p_i^{t+1} - p_i^t) = \sum_{j \in G_i} \left( \lambda_{\mathrm{t},ij+1/2} T_{ij} \left( \Phi_{\mathrm{w},j}^{t+1} - \Phi_{\mathrm{w},i}^{t+1} \right) \right) - \sum_{j \in G_i} \left( \lambda_{\mathrm{n},ij+1/2} T_{ij} \left( \Phi_{\mathrm{c},j}^{t} - \Phi_{\mathrm{c},i}^{t} \right) \right) + (q_\mathrm{n} + q_\mathrm{w}) V_i \tag{4}$$

where $R$ the discrete PDE residual; $G_i$ the connected elements; $T$ is the transmissibility, defined as **Equation 5**:

$$T_{ij} = \frac{A_{ij} k_{ij+1/2}}{d_i + d_j} \tag{5}$$

where $A_{ij}$ is the interface area; $k_{ij+1/2}$ the harmonic means of $k_i$ and $k_j$; $d$ is the vertical distance of grid center to the interface area. The saturation is updated explicitly as **Equation 6**:

$$S_{\mathrm{w},i}^{t+1} = S_{\mathrm{w},i}^{t} + \frac{\sum_{j \in G_i} \left( \lambda_\mathrm{w} T_{ij} \left( \Phi_{\mathrm{w},j}^{t+1} - \Phi_{\mathrm{w},i}^{t+1} \right) \right) \Delta t}{\phi V_i} + \frac{q_\mathrm{w} \Delta t}{\phi} - S_{\mathrm{w},i}^{t} (C_\beta + C_\mathrm{r}) (p_i^{t+1} - p_i^t) \tag{6}$$

In this study, the external boundary conditions are defined as closed boundaries (Neumann boundary conditions with zero velocity). For the IMPES-based numerical method, **Equation 4** will be solved through numerical iterations at each step. The pressure solution is used to explicitly calculate the saturation via **Equation 6**. In our NN-based approach, the mapping $\tilde{p}^{t+1} = f(p^t)$ is established, where the NN undergoes parameter optimization at each timestep to minimize output errors (i.e., enhance physical consistency). This error quantification is defined through a PDE residual-based loss function (**Eq. 7**):

$$L_R = \frac{1}{N_\mathrm{e}} \sum_{i=1}^{N_\mathrm{e}} \left( R(p_i^t, \tilde{p}^{t+1}) \right)^2 \tag{7}$$

The updated NN equation is as follows (**Eq. 8**):

$$\theta^{\tau+1} = \theta^\tau - \eta \nabla L_R \tag{8}$$

where $\theta$ is the NN parameters, and $\eta$ is the learning rate. By iteratively executing **Equation 8** and employing backpropagation with the gradient descent algorithm, we can reduce the loss to the desired threshold, thereby obtaining the pressure solution for each time step. Subsequently, the saturation is calculated through **Equation 6**. This method offers advantages over vanilla PINNs when solving heterogeneous problems: Rather than strictly enforcing PDE-based physical constraints at every discrete point during NN training, the FVM-based approach enforces flux conservation at each discretized grid.

This modification reduces the local steepness of the loss landscape. Moreover, the operation of source/sink terms in **Equation 3** mitigates challenges posed by pressure gradients near well locations.

## 2.2. Tensorization-anchoring Strategy of An-PINN

The mapping process of the EPINN can be expressed as **Equation 9**:

$$f(p^t) = \omega_3 \cdot p^t + (1-\omega_3) \cdot w_3 \cdot \mathrm{ReLU}\big(\omega_2 \cdot w_2 \cdot BN(\omega_1 \cdot (w_1 \cdot p^t + b_1) + m_{\mathrm{adj}}) + b_2\big) + b_3 \tag{9}$$

where, $w_1, w_2, w_3$ and $b_1, b_2, b_3$ are the weights and biases for the upscaling, hidden, and output linear layers, respectively; $BN$ is to the Batch Normalization layer (16); $m_{\mathrm{adj}}$ is the adjacency matrix (35); $\omega_1, \omega_2, \omega_3$ are learnable adaptive parameters, $\omega_1$ adjusts the weight between $m_{\mathrm{adj}}$ and the upscaled $p^t$, $\omega_2$ modifies the adaptive ReLU activation function (17), $\omega_3$ regulates the extent of references to $p^t$ when solving $p^{t+1}$.

Tensorization is a natural choice for enhancing the complexity of NNs, and it has shown impressive results in various state-of-the-art algorithms such as attention mechanisms (32), gating method (36), and the widely adopted Transformer architecture (30). Essentially, it involves alteration at each hidden node. As discussion in studies (2, 9, 16), continuous regularization of intermediate outputs during NN training fundamentally entails ongoing adjustments to the intermediate feature distribution to prevent covariate shift. In this study, we tensorize the adaptive learnable parameters in **Equation 9** to define the Tensorized-EPINN (T-EPINN), thereby modifying **Equation 9** as **Equation 10**:

$$f(p^t) = W_3 \odot p^t + (1-W_3) \odot w_3 \cdot \mathrm{ReLU}\big(W_2 \odot w_2 \cdot BN(W_1 \odot (w_1 \cdot p^t + b_1) + m_{\mathrm{adj}}) + b_2\big) + b_3 \tag{10}$$

where, $W_1, W_2, W_3$ are the tensorized adaptive parameters.

This strategy shares a similar function with the uncertain weighting in (18), enabling the NN to autonomously generate an appropriate fusion weight for different nodes. This approach aligns more closely with gating-based methods rather than attention-based mechanisms, as the weight values are primarily adaptively learned by the NN through backpropagation instead of being computed through specific rules during forward propagation. Tensorization operations could provide enhanced coupling between these different modules, improving their collaborative performance: **1)** Deeper layers are generally considered to learn more detailed features, while shallower layers capture more macroscopic features. These layers may contribute differently to the local and global computational accuracy of physical fields. **2)** Adaptive components at the adjacency matrix embedding and activation function modules involve processing high-dimensional hidden matrices, which introduce more parameters and handles complex patterns. In contrast, the gating module operates on tensors that have been projected back to the original pressure field dimensions through linear transformations, exerting a more direct influence on prediction outcomes.

However, it is evident that the tensorization increases the computational resource consumption, especially when manipulating large-scale tensors. Numerous efforts (14, 15, 25) have been made to compress NNs to enhance both training and inference speeds. We designed the tensorization-anchoring strategy inspired by the fine-tuning strategy from transfer learning (41) and continual learning (12). In our simulations, the inputs and outputs of the NN correspond to physically meaningful quantities—specifically, the spatial distribution of the pressure field $p$. To ensure consistency, we applied a standard normalization criterion for pressure (**Eq. 11**):

$$p_{scal} = \frac{p}{p_{init}} \tag{11}$$

where $p_{init}$ is the initial pressure of the reservoir formation and $p_{scal}$ is the normalized pressure. **Equation 11** uniformly scales the pressure to a value close to 1. In subsequent time steps, the pressure will also vary in the vicinity of 1, and neural networks excel at learning on this scale. Consequently, the inputs and outputs of the NN inherently possess similar distribution, this ensures the feature distribution will not exhibit pronounced changes. In the early stages, learnable weights are generated in $W_1, W_2, W_3$ during the NN solving process to alter the features across different nodes. As the NN's learning

approaches stable convergence, these learnable parameters are detached from the computational graph, thereby anchoring the hidden nodes to fixed distribution states. Specifically, after completing the training phase of the tensorizing epochs (Ta), we perform the anchoring operation: the latest NN parameters are output, and the parameters of the adaptive components are fixed to fully detach them from the computational graph. Training is then restarted. In our experiments, this procedure typically reduces GPU memory usage by 30%–50%. Compared to conventional pruning operations, this methodology fundamentally eliminates the fixed components from both forward and backward propagation computational graphs, rather than merely deactivating connections. The detailed workflow of the An-EPINN framework is outlined in **Algorithm 1**.

| **Algorithm 1:** The training algorithm for An-PINN. | |
|---|---|
| 1: | inputs: simulation time $T$, time step $\Delta t$, tolerance $\varepsilon$, and maximum epoch $M$, learning rate $lr$ |
| 2: | set up Adam (19) with $lr$, set up tensorizing epoch $T_a$ |
| 3: | **while** $t < T_a$ **do** |
| 3: | if $t$ = first time: initialize NN by using Xavier (20) method, and input IC |
| 4: | if $t$ > first time: initialize NN by using the trained NN of previous time step |
| 5: | **loop1** using $\Delta t$, until $\varepsilon$ or $M$ is reached, retain the result associated with the lowest loss |
| 6: | **explicit update** other parameters, set $t = t + 1$ |
| 6: | end while |
| 7: | remove the $W_1$, $W_2$ and $W_3$ from the computation graph, reset Adam with $\boldsymbol{lr}$ |
| 8: | **while** $t < T$ do |
| 9: | initialize with the NN of the previous time step, input IC |
| 10: | **loop1** using Δt, until $\varepsilon$ or $M$ is reached, retain the result associated with the lowest loss |
| 11: | **explicit update** other parameters, set $t = t + 1$ |
| 12: | **end while** |

# 3. RESULTS AND DISCUSSIONS

## 3.1. Model Setting

We conducted tests on both homogeneous and heterogeneous reservoir cases. The grid size for the homogeneous case was 25×25, with each grid measuring 5m×5m. The absolute permeability was 10 mD, and the relative permeability curves are shown in **Figure 1**. Water was defined as the wetting phase and oil as the non-wetting phase. The porosity was set to 0.25, the initial reservoir pressure was 25MPa, and the rock compressibility was 1.0×10-8 $Pa^{-1}$. The initial water saturation was 0.2, the irreducible water saturation was 0.2, water density was 1000 $kg \bullet m^{-3}$, water viscosity was 1.0 $mPa \bullet s$, and water compressibility was 1.0×10-10 $Pa^{-1}$. The residual oil saturation was 0.2, oil density was 800 $kg \bullet m^{-3}$, oil viscosity was 1.8 $mPa \bullet s$, and oil compressibility was 1.0×10-10 $Pa^{-1}$. One injection well and one production well were positioned at opposite corners of the reservoir diagonal, with the injection and production rate set to 2.0 $m3 \bullet d^{-1}$. The wellbore radius was 0.1, and the skin factor was 0.0. The

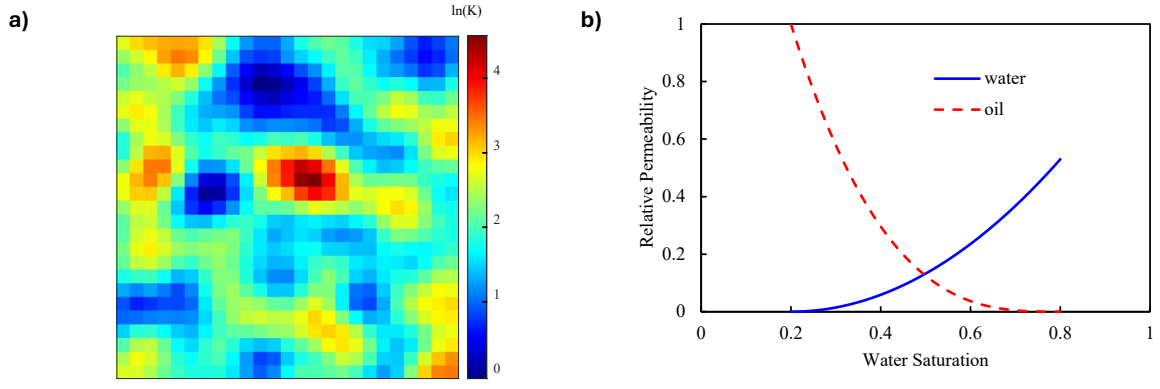


**Figure 1:** The heterogeneous permeability distribution **(a)** and relative permeability curve **(b)**.

heterogeneous reservoir example had the same basic parameters as the homogeneous case except for the absolute permeability; the heterogeneous permeability distribution varied from 1.0 mD to 100 mD as shown in **Figure 1a**. All models simulated 1000 days, with a timestep of 0.01 days. All reference solutions used for comparison were generated by an in-house precise simulator (34).

**Table 1:** The training speed of different neural networks.

| Case | Method | Iteration epochs | Elapsed time |
|---|---|---|---|
| Homogeneous | EPINN | 509104 | 2976.73s |
| | T-EPINN | 338062 | 3572.14s |
| | An-EPINN | 339204 | 2039.97s |
| Heterogeneous | EPINN | 581002 | 3397.12s |
| | T-EPINN | 409612 | 4328.18s |
| | An-EPINN | 407173 | 2448.74s |

Three types of NNs, EPINN (without tensorization and anchoring), T-EPINN (with tensorization only), and An-EPINN (with both tensorization and anchoring), were set up for solving the problem. The total number of learnable parameters for EPINN was 394,379, while for T-EPINN and An-EPINN, it was 1,176,251, of which 781,875 parameters in An-EPINN could be detached from the computational graph. Based on experimental observations and guided by the methodology in (35), we manually adjusted and selected hyperparameters during the experiments to keep the models near their respective relative optima. The initial values of $w_1, w_2, w_3$ were set to 0.01, 0.5, 0.1, respectively; initial values in tensors $W_1, W_2$, and $W_3$ were uniformly set to 0.01, 0.5, and 0.9, respectively; the learning rate was set at 0.01; the maximum number of iterations per time step was 1000, and the residual convergence threshold was $1.0 \times 10^{-18}$. Notably, in our practical testing, after approximately 100 epochs of iteration, the NNs tended to stabilize in convergence, and the adaptive parameters approached saturation. Consequently, the tensorizing epoch Ta for An-EPINN was set to 100. Additionally, it is worth noting that early or premature parameter fixing during training had no significant impact on solution accuracy, as convergence thresholds were strictly enforced at each time step. However, such premature fixing (before stabilized convergence) could hinder the model's ability to capture general patterns, thereby requiring more

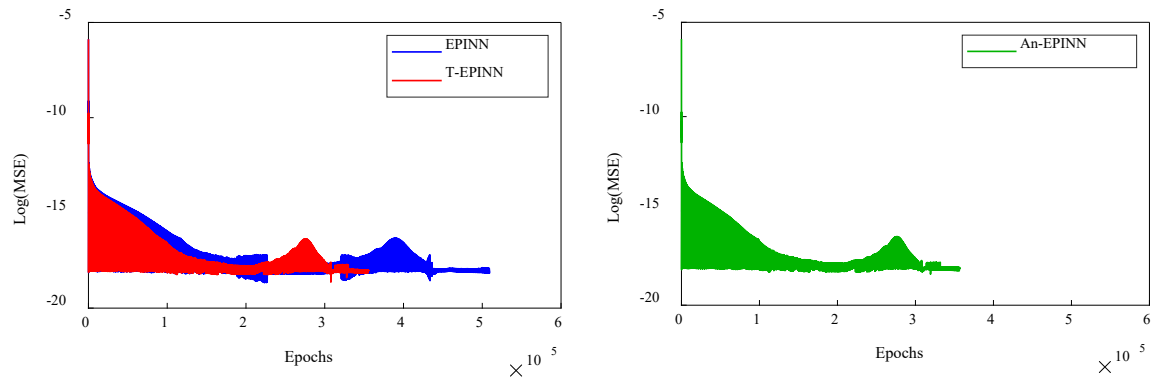


**Figure 2:** The training speed of different neural networks in the homogeneous case.

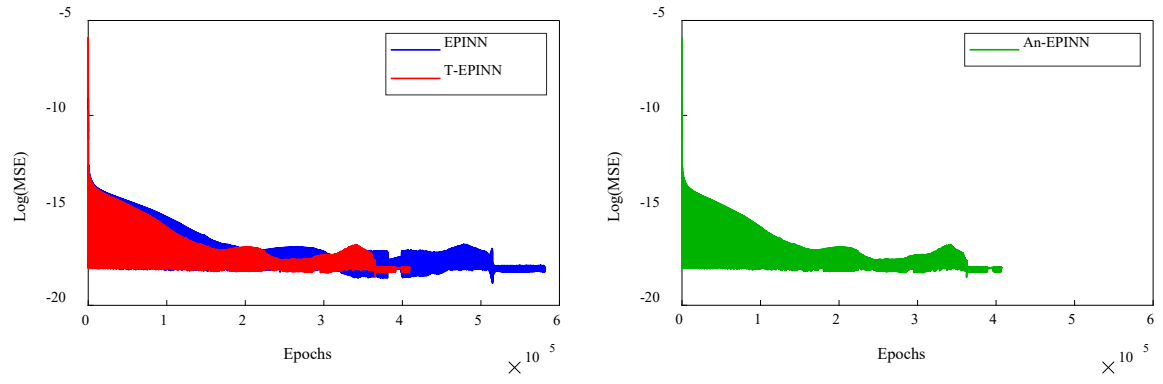


**Figure 3:** The training speed of different neural networks in the heterogeneous case.

 https://doi.org/10.69631/ipj.v2i3nr67

iterations per time step. These hyperparameter configurations represent a general performance in our experiments. While further fine-tuning could marginally improve model performance, such adjustments were insufficient to alter the conclusions of this study. All NNs were built and trained on the PyTorch framework (Version 1.12.1) (26), with all learning executed on an Nvidia 4060 GPU.

## 3.2. Results of NN Training

As shown in **Figure 2** and **Figure 3**, under both the homogeneous and heterogeneous scenario, T-EPINN consistently demonstrated a faster convergence speed than EPINN after tensorization, whereas the convergence rate for An-EPINN was similar to that of T-EPINN. **Table 1** reveals that although T-EPINN required fewer epochs, its elapsed time in both homogeneous and heterogeneous cases was higher than EPINN. By detaching the adaptive part from the computational graph, An-EPINN significantly reduced

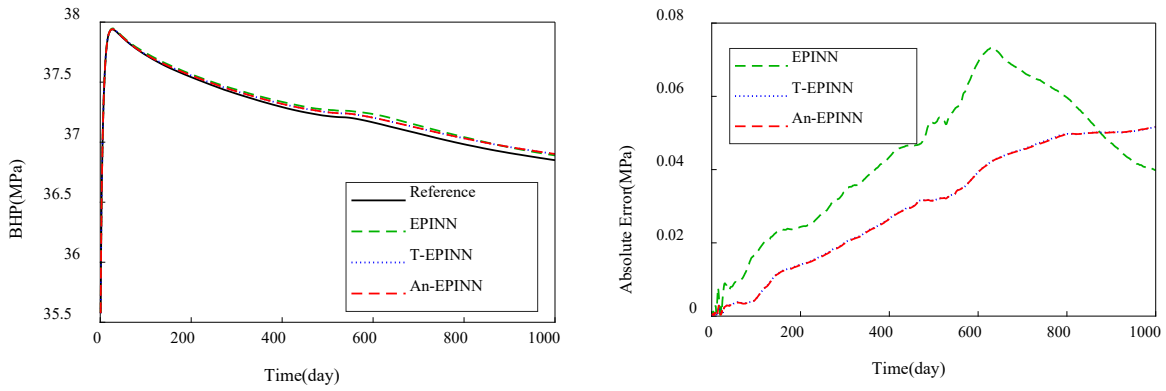


**Figure 4:** The injection well BHP of different neural networks in the homogeneous case.

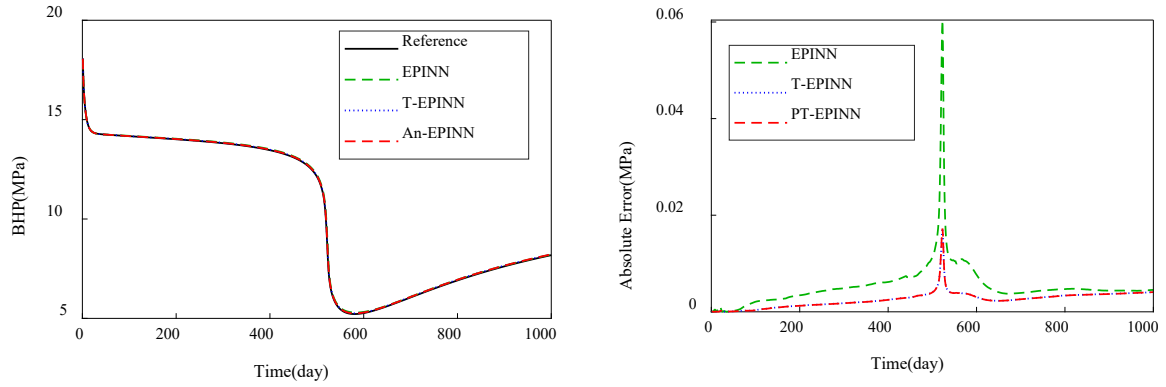


**Figure 5:** The production well BHP of different neural networks in the homogeneous case.

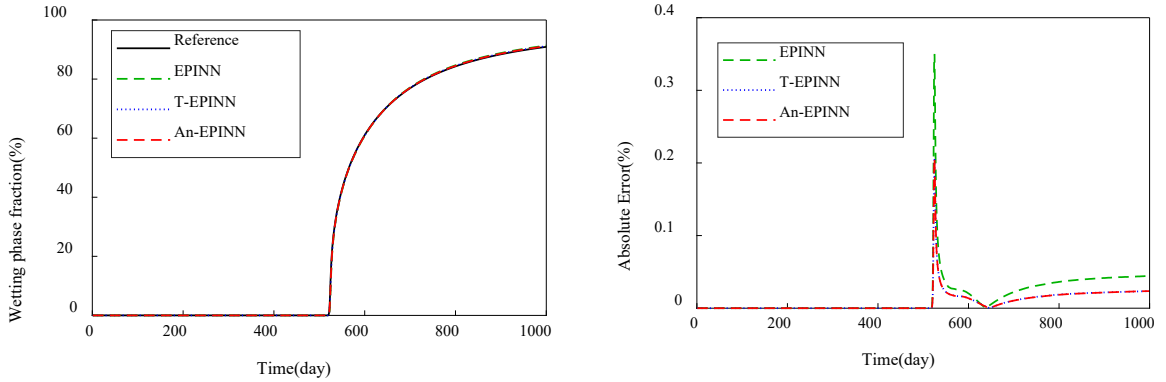


**Figure 6:** The wetting phase fraction of different neural networks in the homogeneous case.

 https://doi.org/10.69631/ipj.v2i3nr67

its elapsed time, with a reduction of approximately 31.47% in the homogeneous case and 27.91% in the heterogeneous case.

## 3.3. Results of injection and Production Curves

In **Figure 4**, **Figure 5**, and **Figure 6**, under homogeneous conditions, the convergence accuracy of T-EPINN and An-EPINN was comparable (with injection BHP errors below 0.06 MPa, production BHP errors under 0.02 MPa, and water fraction errors less than 0.3%) and outperformed EPINN (with injection BHP errors below 0.08 MPa, production BHP errors under 0.06 MPa, and water fraction errors less than 0.4%). In **Figure 7**, **Figure 8**, and **Figure 9**, under heterogeneous conditions, the calculated injection BHP and wetting phase fraction from T-EPINN and An-EPINN remained highly consistent (injection BHP errors below 0.08 MPa; water fraction errors under 0.4% except for a brief period before water breakthrough),

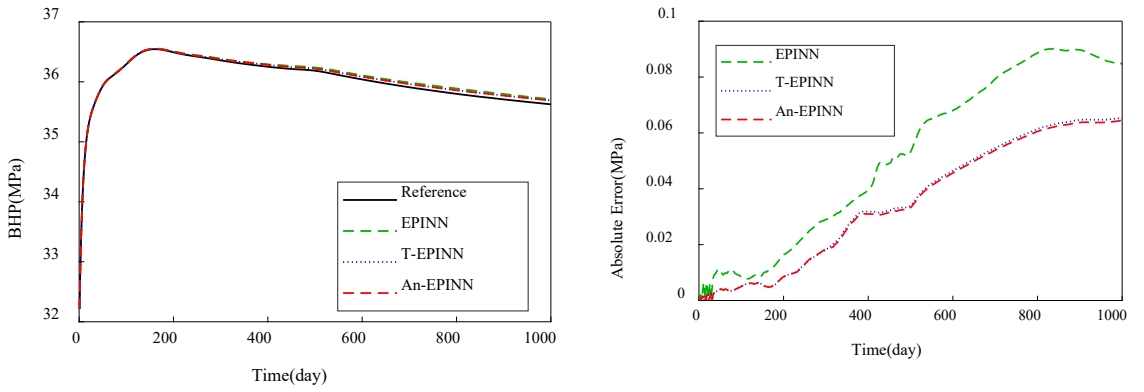


**Figure 7:** The injection well BHP of different neural networks in the heterogeneous case.

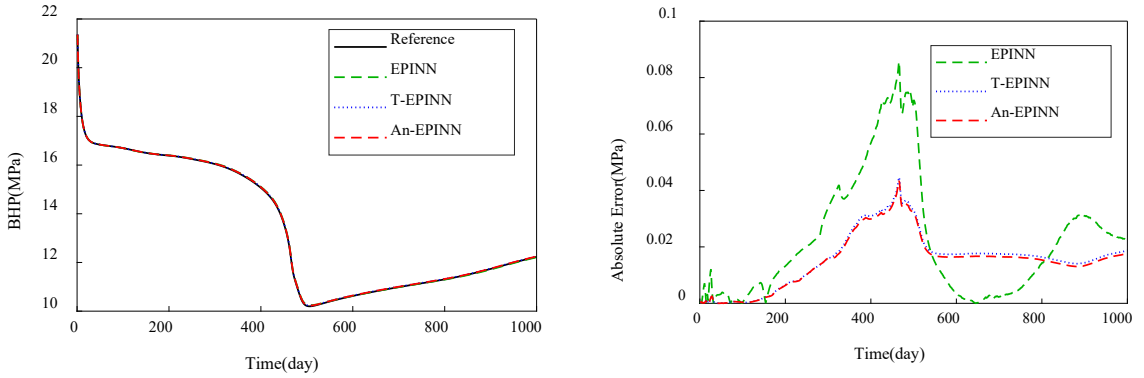


**Figure 8:** The production well BHP of different neural networks in the heterogeneous case.

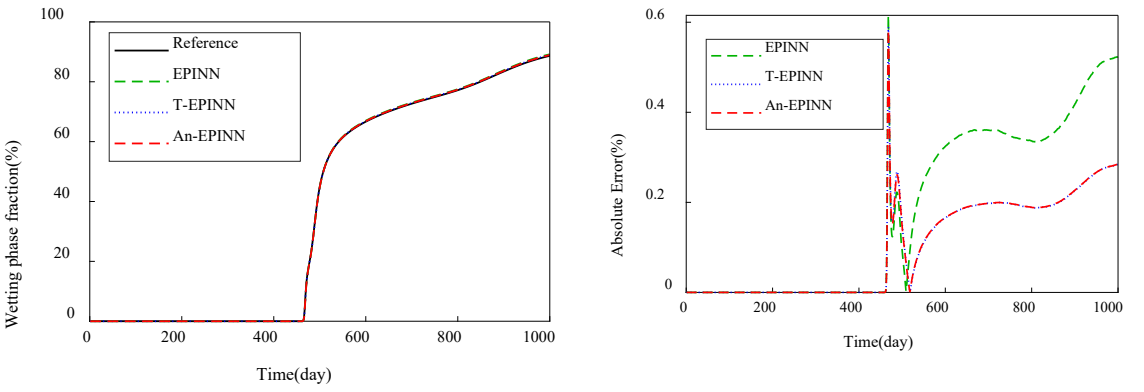


**Figure 9:** The wetting phase fraction of different neural networks in the heterogeneous case.

 https://doi.org/10.69631/ipj.v2i3nr67

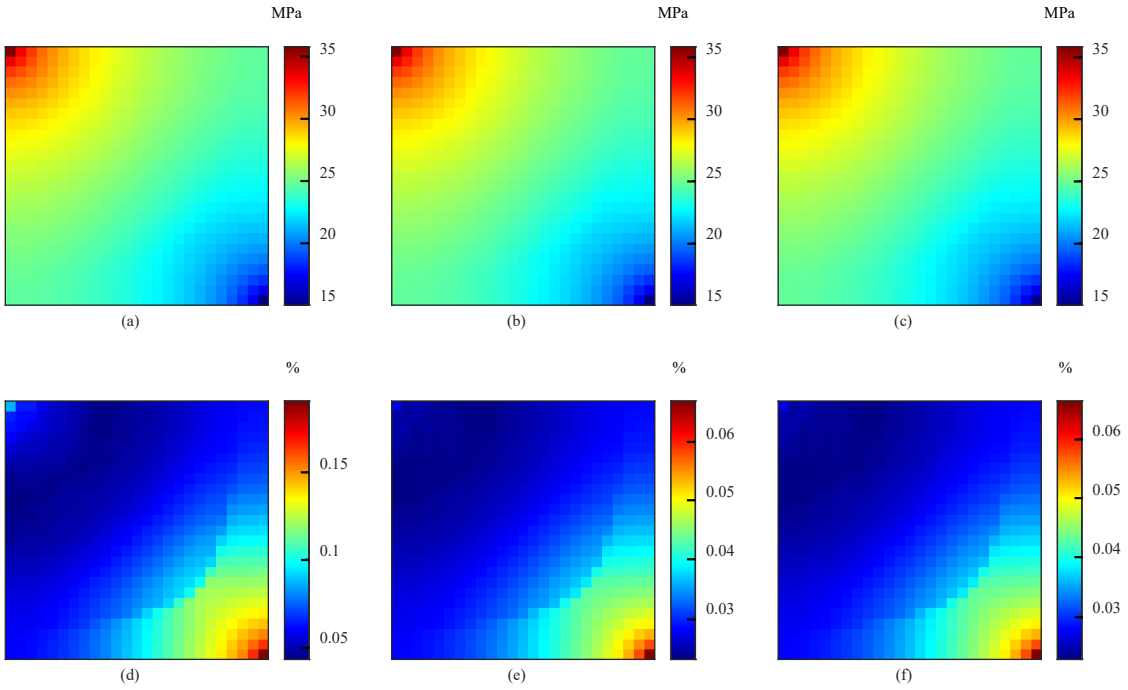


**Figure 10:** The pressure distribution of the homogeneous case at 500 days; **a)** EPINN pressure, **b)** T-EPINN pressure, **c)** An-EPINN pressure, **d)** EPINN error, e) T-EPINN error, **f)** An-EPINN error.

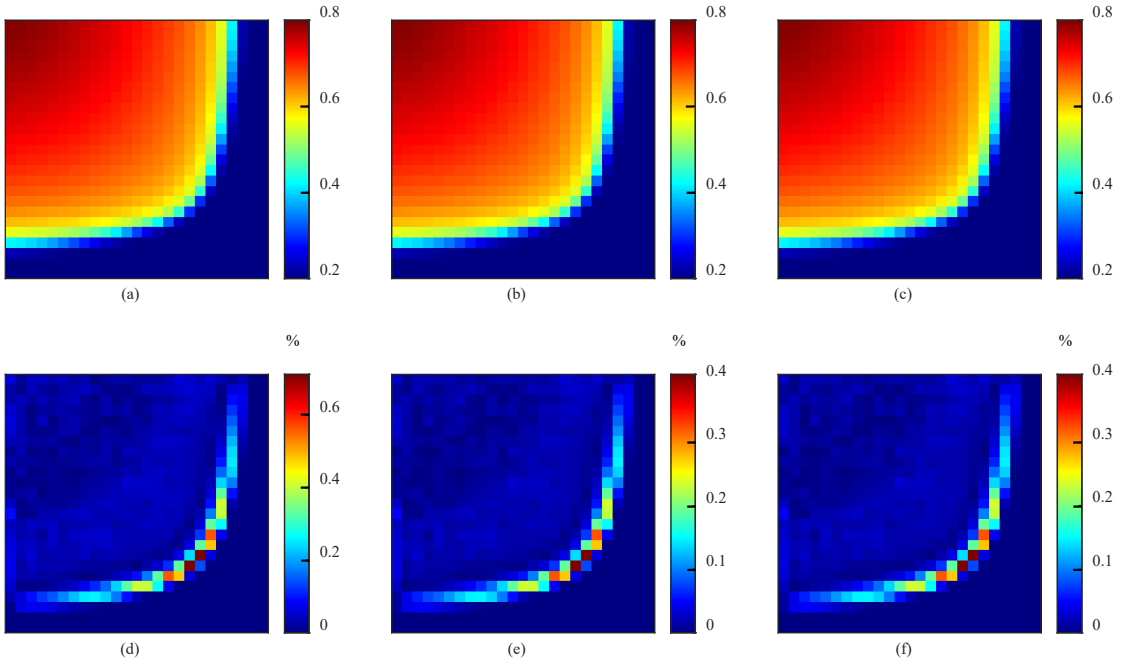


**Figure 11:** The water saturation distribution of the homogenous case 500 days; **a)** EPINN saturation, **b)** T-EPINN saturation, **c)** An-EPINN saturation, **d)** EPINN error, e) T-EPINN error, **f)** An-EPINN error.

and exhibited higher accuracy than EPINN (injection BHP errors below 0.1 MPa, water fraction errors under 0.6%). The production well BHP calculations from T-EPINN and An-EPINN (errors below 0.06 MPa) were more precise than those from EPINN (errors up to 0.1 MPa) for most of the simulation period, except for a short period of post-water-breakthrough. Overall, T-EPINN and An-EPINN achieved nearly identical computational accuracy, with both slightly surpassing EPINN in overall performance. This indicates that tensorization operations enabled the NNs to learn more generalized patterns, while the anchoring technique maintained precision despite reduced parameter updating.

 https://doi.org/10.69631/ipj.v2i3nr67

## 3.4. Results of flow fields

As shown in **Figure 10**, the maximum relative errors in the pressure field obtained from T-EPINN and An-EPINN were both below 0.07%, demonstrating improved performance compared to EPINN (>0.15%). In **Figure 10d-f**, T-EPINN and An-EPINN exhibited relatively lower maximum relative errors near the injection well (top left region). **Figure 11** shows that the errors of T-EPINN and An-EPINN were comparable (<0.4%) and lower than those of EPINN (>0.6%). Under heterogeneous conditions (**Fig. 12**, **Fig. 13**), the maximum relative errors in saturation and pressure fields for both T-EPINN and An-EPINN (homogeneous case: <0.13%; heterogeneous case: <0.4%) were substantially smaller than EPINN's

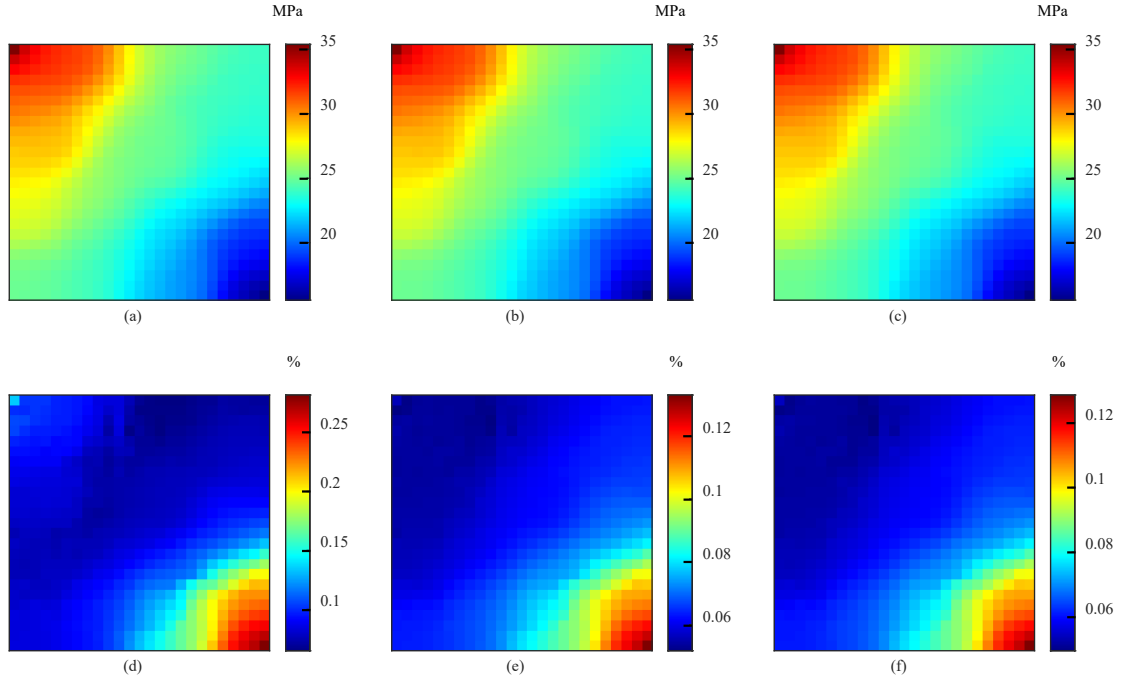


**Figure 12:** The pressure distribution of the heterogeneous case at 500 days; **a)** EPINN pressure, **b)** T-EPINN pressure, **c)** An-EPINN pressure, **d)** EPINN error, **e)** T-EPINN error, **f)** An-EPINN error.

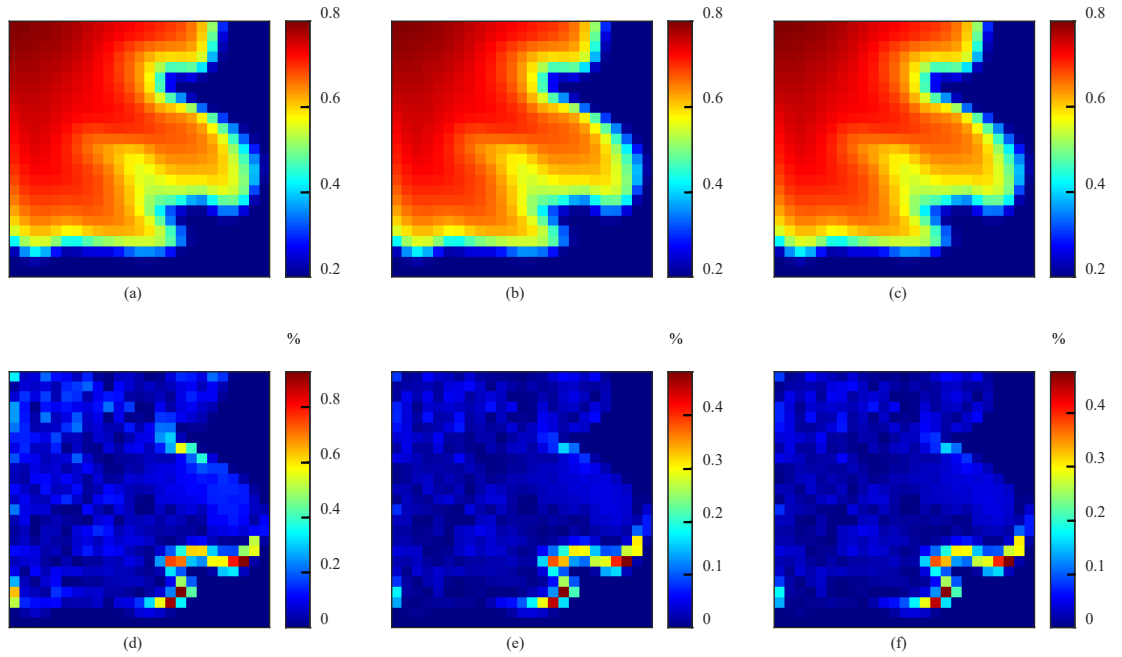


**Figure 13:** The saturation distribution of the heterogeneous case at 500 days; **a)** EPINN saturation, **b)** T-EPINN saturation, **c)** An-EPINN saturation, **d)** EPINN error, **e)** T-EPINN error, **f)** An-EPINN error.

 https://doi.org/10.69631/ipj.v2i3nr67

results (homogeneous case: >0.25%; heterogeneous case: >0.8%). Comparative analysis of subplots (e-f) versus (d) in **Figure 12** and **Figure 13** reveals that T-EPINN and An-EPINN achieved reduced computational errors in proximity to the injection well. In summary, An-EPINN demonstrates convergence accuracy equivalent to T-EPINN and superior to conventional EPINN methodology. The improved near wellbores accuracy suggests the enhanced capability to capture high-gradient flow patterns, thereby improving NN performance.

## 4. CONCLUSIONS

In this study, we proposed a novel Anchored-PINN (An-EPINN) to improve the solving of two-phase flow in porous media. Through the "tensorization-anchoring" strategy, a multitude of learnable parameters in the tensorized adaptive adjacency embedding, adaptive activation functions, and adaptive gating are anchored to a fixed hidden state. This algorithm can leverage the adaptive mechanisms to accelerate NN convergence and avoid the redundant calculation during backpropagation. In both homogeneous and heterogeneous scenarios, the tensorized operations significantly enhance the NN's convergence accuracy, the An-EPINN achieves superior computational speeds while maintaining improved convergence accuracy. This study offers viable approaches for further enhancing the convergence accuracy while reducing the computational resource consumption. It is valuable for running NN solutions on personal devices with limited computational resources and is expected to play a greater role in solving larger-scale problems.

However, there remains room for further improvement in the current research: the initial stage of the tensorization-anchoring workflow still requires large-scale matrix operations. Future investigations plan to explore matrix factorization or dimensionality reduction approaches to substantially reduce computational complexity in this phase. While the present study was confined to 2D structured grids, we plan on extending the methodology to 3D reservoir simulations, unstructured grids and additional PDE types. Furthermore, the current research primarily revolves around the EPINN framework. In the future, we will extend this work to more state-of-the-art PINN variants and conduct systematic comparisons with existing methodologies.

## STATEMENTS AND DECLARATIONS

### Supplementary Material

All supplementary or supporting information provided by the authors will be published online alongside the main article. When such material is available, it will also be indicated here along with a link to the pdf for downloading.

### Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 52474067, 52325402, 52274057 and 52074340), the National Key R&D Program of China (No. 2023YFB4104200), the Major Scientific and Technological Projects of CNOOC (No. CCL2022RCPS0397RSN), 111 Project (No. B08028), Youth Innovation and Technology Support Program for Higher Education Institutions of Shandong Province, China (No. 2022KJ070).

### Author Contributions

**Jingqi Lin:** Conceptualization, Investigation, Methodology, Validation, Visualization, Writing – original draft. **Xia Yan:** Conceptualization, Data curation, Funding acquisition, Investigation, Resources, Software, Writing – original draft. **Kai Zhang:** Formal analysis, Funding acquisition, Project administration, Resources, Supervision, Writing – review & editing. **Zhao Zhang:** Data curation, Project administration, Resources, Software, Supervision, Writing – review & editing. **Jun Yao:** Formal analysis, Project administration, Resources, Supervision, Writing – review & editing.

### Conflicts of Interest

The authors have no conflicts of interest to declare.

## Data, Code & Protocol Availability

All study resources are available from the corresponding author upon reasonable request.


## Funding Received

National Natural Science Foundation of China (No. 52474067, 52325402, 52274057 and 52074340), the National Key R&D Program of China (No. 2023YFB4104200), the Major Scientific and Technological Projects of CNOOC (No. CCL2022RCPS0397RSN), 111 Project (No. B08028), Youth Innovation and Technology Support Program for Higher Education Institutions of Shandong Province, China (No. 2022KJ070)



## ORCID IDs

Jingqi Lin https://orcid.org/0009-0001-4860-4036
Kai Zhang https://orcid.org/0000-0002-6691-6762
Zhao Zhang https://orcid.org/0000-0003-0523-5071
Jun Yao https://orcid.org/0000-0002-6720-584X


## REFERENCES


1. Anagnostopoulos, S. J., Toscano, J. D., Stergiopulos, N., & Karniadakis, G. E. (2024). Residual-based attention in physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421, 116805. https://doi.org/10.1016/j.cma.2024.116805
2. Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization (No. arXiv:1607.06450). arXiv. https://doi.org/10.48550/arXiv.1607.06450
3. Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2017). Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, 18(1), 5595–5637.
4. Brunton, S. L., & Kutz, J. N. (2024). Promising directions of machine learning for partial differential equations. *Nature Computational Science*, 4(7), 483–494. https://doi.org/10.1038/s43588-024-00643-2
5. Cai, S., Mao, Z., Wang, Z., Yin, M., & Karniadakis, G. E. (2021). Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12), 1727–1738. https://doi.org/10.1007/s10409-021-01148-1
6. Cen, J., & Zou, Q. (2024). Deep finite volume method for partial differential equations. *Journal of Computational Physics*, 517, 113307. https://doi.org/10.1016/j.jcp.2024.113307
7. Chen, C. L. P., & Liu, Z. (2018). Broad learning system: An effective and efficient incremental learning system without the need for deep architecture. *IEEE Transactions on Neural Networks and Learning Systems*, 29(1), 10–24. https://doi.org/10.1109/TNNLS.2017.2716952
8. Chen, H., Kou, J., Sun, S., & Zhang, T. (2019). Fully mass-conservative IMPES schemes for incompressible two-phase flow in porous media. *Computer Methods in Applied Mechanics and Engineering*, 350, 641–663. https://doi.org/10.1016/j.cma.2019.03.023
9. Chen, Z., Badrinarayanan, V., Lee, C.-Y., & Rabinovich, A. (2017). Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. https://doi.org/10.48550/ARXIV.1711.02257
10. Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., & Piccialli, F. (2022). Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3), 88. https://doi.org/10.1007/s10915-022-01939-z
11. Daw, A., Karpatne, A., Watkins, W., Read, J., & Kumar, V. (2017). Physics-guided neural networks (PGNN): An application in lake temperature modeling. arXiv. https://doi.org/10.48550/ARXIV.1710.11431
12. Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., et al. (2021). A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1. https://doi.org/10.1109/TPAMI.2021.3057446
13. Eymard, R., Gallouët, T., & Herbin, R. (2000). Finite volume methods. In *Handbook of Numerical Analysis* (Vol. 7, pp. 713–1018). Elsevier. https://doi.org/10.1016/S1570-8659(00)07005-8
14. Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6), 1789–1819. https://doi.org/10.1007/s11263-021-01453-z
15. Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv. https://doi.org/10.48550/ARXIV.1510.00149

16. Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv. https://doi.org/10.48550/ARXIV.1502.03167
17. Jagtap, A. D., Kawaguchi, K., & Karniadakis, G. E. (2020). Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404, 109136. https://doi.org/10.1016/j.jcp.2019.109136
18. Kendall, A., Gal, Y., & Cipolla, R. (2017). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. arXiv. https://doi.org/10.48550/ARXIV.1705.07115
19. Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization (No. arXiv:1412.6980). arXiv. https://doi.org/10.48550/arXiv.1412.6980
20. Kumar, S. K. (2017). On weight initialization in deep neural networks (No. arXiv:1704.08863). arXiv. https://doi.org/10.48550/arXiv.1704.08863
21. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444. https://doi.org/10.1038/nature14539
22. Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2021). Fourier neural operator for parametric partial differential equations (No. arXiv:2010.08895). arXiv. https://doi.org/10.48550/arXiv.2010.08895
23. Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E. (2021). Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3), 218–229. https://doi.org/10.1038/s42256 -021-00302-5
24. Moortgat, J., & Firoozabadi, A. (2013). Three-phase compositional modeling with capillarity in heterogeneous and fractured media. *SPE Journal*, 18(06), 1150–1168. https://doi.org/10.2118/159777-PA
25. Novikov, A., Podoprikhin, D., Osokin, A., & Vetrov, D. (2015). Tensorizing neural networks. arXiv. https://doi.org/10.48550/ARXIV.1509.06569
26. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. arXiv. https://doi.org/10.48550/ARXIV.1912.01703
27. Peaceman, D. W. (1993). Representation of a horizontal well in numerical reservoir simulation. *SPE Advanced Technology Series*, 1(01), 7–16. https://doi.org/10.2118/21217-PA
28. 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. *Journal of Computational Physics*, 378, 686–707. https://doi.org/10.1016/j.jcp.2018.10.045
29. Uriarte, C., Pardo, D., & Omella, Á. J. (2022). A finite element based deep learning solver for parametric PDES. *Computer Methods in Applied Mechanics and Engineering*, 391, 114562. https://doi.org/10.1016/j.cma.2021.114562
30. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., et al. (2017). Attention is all you need. arXiv. https://doi.org/10.48550/ARXIV.1706.03762
31. Wang, N., Chang, H., & Zhang, D. (2022). Surrogate and inverse modeling for two-phase flow in porous media via theory-guided convolutional neural network. *Journal of Computational Physics*, 466, 111419. https://doi.org/10.1016/j.jcp.2022.111419
32. Woo, S., Park, J., Lee, J.-Y., & Kweon, I. S. (2018). CBAM: Convolutional block attention module (No. arXiv:1807.06521). arXiv. https://doi.org/10.48550/arXiv.1807.06521
33. Wu, C., Zhu, M., Tan, Q., Kartha, Y., & Lu, L. (2023). A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403, 115671. https://doi.org/10.1016/j.cma.2022.115671
34. Yan, X., Huang, Z., Yao, J., Li, Y., & Fan, D. (2016). An efficient embedded discrete fracture model based on mimetic finite difference method. *Journal of Petroleum Science and Engineering*, 145, 11–21. https://doi.org/10.1016/j.petrol.2016.03.013
35. Yan, X., Lin, J., Wang, S., Zhang, Z., Liu, P., et al. (2024). Physics-informed neural network simulation of two-phase flow in heterogeneous and fractured porous media. *Advances in Water Resources*, 189, 104731. https://doi.org/10.1016/j.advwatres.2024.104731
36. Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. *Neural Computation*, 31(7), 1235–1270. https://doi.org/10.1162/neco_a_01199
37. Zhang, B. (2023). Nonlinear mode decomposition via physics-assimilated convolutional autoencoder for unsteady flows over an airfoil. *Physics of Fluids*, 35(9), 095115. https://doi.org/10.1063/5.0164250

38. Zhang, Z. (2022). A physics-informed deep convolutional neural network for simulating and predicting transient Darcy flows in heterogeneous reservoirs without labeled data. *Journal of Petroleum Science and Engineering*, 211, 110179. https://doi.org/10.1016/j.petrol.2022.110179
39. Zhang, Z., Yan, X., Liu, P., Zhang, K., Han, R., & Wang, S. (2023). A physics-informed convolutional neural network for the simulation and prediction of two-phase Darcy flows in heterogeneous porous media. *Journal of Computational Physics*, 477, 111919. https://doi.org/10.1016/j.jcp.2023.111919
40. Zhao, C., Zhang, F., Lou, W., Wang, X., & Yang, J. (2024). A comprehensive review of advances in physics-informed neural networks and their applications in complex fluid dynamics. *Physics of Fluids*, 36(10), 101301. https://doi.org/10.1063/5.0226562
41. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2021). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1), 43–76. https://doi.org/10.1109/JPROC.2020.3004555