TIME-SPLITTING FOURIER NEURAL OPERATOR WITH COORDINATE INJECTION FOR SCALABLE RESERVOIR SIMULATION

Gabriel F. Barros[1], **Amanda C.N. de Oliveira**[2], **Rômulo M. Silva**[1], **Ezequiel S. Santos**[1], **Rodolfo S. M. Freitas**[3], **Fernando A. Rochinha**[3] **& Alvaro L. G. A. Coutinho**[1]
1. Department of Civil Engineering
2. Department of Systems Engineering and Computer Science
3. Department of Mechanical Engineering
COPPE - Federal University of Rio de Janeiro
Rio de Janeiro, RJ, Brazil
{gabriel.barros, romulo.silva, ezequiel.souza, alvaro}@coc.ufrj.br
amandacno@cos.ufrj.br
{rodolfosmfreitas, faro}@mecanica.coppe.ufrj.br

Dakshina M. Valiveti & Xiao-Hui Wu
ExxonMobil Technology and Engineering Company
Spring, TX, USA
{dakshina.m.valiveti, xiao-hui.wu}@exxonmobil.com

ABSTRACT

In this work, we propose a Time-Splitting FNO with Coordinate Injection that partitions the temporal domain into manageable sub-windows, reducing training memory requirements regarding the time horizon. To mitigate the loss of global context inherent to splitting, we introduce an explicit time-coordinate injection mechanism that breaks the shift-invariance of the spectral operator, allowing the network to learn non-stationary dynamics. We validate our approach on the SPE10 benchmark. Results demonstrate that our method reduces peak GPU memory usage by approximately 7 times while maintaining competitive accuracy on key quantities of interest, such as oil production rates. The proposed time-splitting strategy is a promising way to reduce memory consumption in long-horizon spatio-temporal FNOs while preserving useful predictive quality on the tested benchmark.

1 INTRODUCTION

Numerical simulations are a vital decision-making tool in reservoir engineering (Samnioti & Gaganis, 2023) but the requirement for fine meshes and complex models yields accurate solutions at substantial computational costs, making many-query applications (such as uncertainty quantification and optimization) prohibitive when relying solely on numerical solvers. Scientific Machine Learning (SciML) surrogate models have emerged as an efficient alternative to emulate reservoir simulators for these computationally intensive studies (Bocoum & Rasaei, 2023; Badawi & Gildin, 2025). A breakthrough in SciML is the development of Neural Operators (NOs) (Kovachki et al., 2023), specifically the Fourier Neural Operator (FNO) (Li et al., 2020). Despite being generalizable and yielding good results in multiphase porous media flow, the usual FNO input tensor for a transient 3D PDE takes the high-dimensional form $(N_s, N_{c_{in}}, N_t, N_x, N_y, N_z)$, where N_s is the number of samples, $N_{c_{in}}$ is the number of input channels, N_t is the temporal dimension and N_x, N_y, N_z are the spatial dimensions. This leads to significant memory consumption when scaling to three-dimensional high-resolution problems (Grady et al., 2023; Liu et al., 2023). Addressing this memory bottleneck is critical for applying FNOs to realistic reservoir scales.

Current approaches to handling temporal dynamics in FNOs typically fall into two extremes: spatio-temporal unified (STU) formulations or autoregressive (AR) schemes (Ye et al., 2025). Unified approaches treat time as an additional spatial coordinate, benefiting from global consistency but suffering from rapidly growing memory footprints and fixed temporal discretization. Conversely, AR methods reduce memory requirements by propagating solutions sequentially but are prone to error accumulation and stability issues over long horizons (McCabe et al., 2023). An intermediate time-as-a-parameter approach treats time as an explicit input coordinate, allowing for direct querying of arbitrary temporal locations and super-resolution without enforcing sequential dependence (Ye et al., 2025).

In this study, we propose a time-as-a-parameter FNO strategy for multiphase flow where the temporal domain $\mathcal{T}$ is split into smaller uniform subdomains $\mathcal{T}_i^c$ such that $\mathcal{T} = \cup\mathcal{T}_i^c$. By dividing the samples with total time horizon containing N_t snapshots into chunks of N_t^c snapshots and applying 4D spectral convolution, this method accommodates the fine spatial discretizations and long production histories of real-life reservoirs without exceeding available GPU memory. Section 2 details this methodology and its mathematical foundations; Section 3 presents the physical model and core results; and Section 4 offers concluding remarks, with further technical discussions provided in Appendices A, B, and C.

2 METHOD

We build upon the general Fourier Neural Operator (FNO) architecture, detailed in Appendix A. While effective, the standard 4D FNO requires processing the entire spatio-temporal history simultaneously, creating a memory bottleneck. To address this, we propose a Time-Splitting strategy that partitions the global temporal domain $\mathcal{T}$ into a sequence of N_{splits} sub-intervals $\mathcal{T}_i^c = [t_i, t_{i+1}]$. This restricts the spectral convolutions to manageable windows, reducing the memory complexity from $\mathcal{O}(N_t)$ to $\mathcal{O}(N_t/N_{splits})$ per forward pass. That is, the usual input tensor for FNO would be now of dimensions $(N_s, N_{c_{in}}, N_t^c, N_x, N_y, N_z)$, where $N_t^c < N_t$.

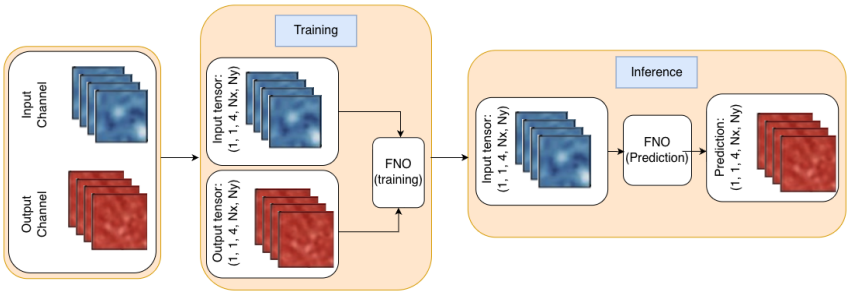
A critical challenge with pure time-splitting is the loss of global temporal context. Standard FNO kernels are shift-invariant, meaning the operator G_θ would process an early interval $[t_0, t_1]$ identically to a late interval $[t_k, t_{k+1}]$ if the local features were similar. To resolve this ambiguity, we introduce a Coordinate Injection mechanism (Mildenhall et al., 2020; Tancik et al., 2020). We break the shift-invariance by explicitly augmenting the input feature space with the absolute time coordinate. The input to the first layer is reformulated as an augmented vector field $\tilde{\mathbf{v}}_{in} : \mathcal{D}_i^c \to \mathbb{R}^{d_v+1}$:

$$\tilde{\mathbf{v}}_{in}(\mathbf{x}, t) = (\mathbf{v}_{in}(\mathbf{x}, t),\, \lambda(t)) \quad \text{for } t \in \mathcal{T}_i^c, \tag{1}$$

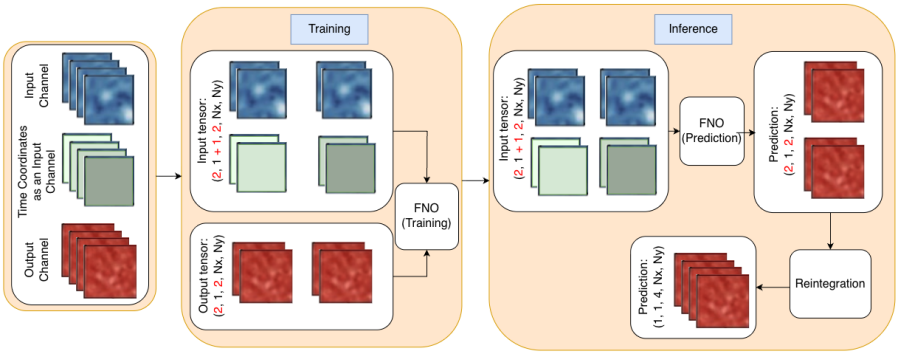
where $\lambda(t)$ is the coordinate embedding of the absolute time and $\mathbf{v}_{in}(\mathbf{x}, t)$ are the original input channels considered in the FNO modeling. Although this idea is not entirely new (Liu et al., 2018), its usage as a strategy to mitigate memory overhead on FNOs when scaling to real world problems was neither deeply assessed in terms of efficiency nor performance. By injecting absolute time, the operator becomes effectively non-stationary, enabling the learned kernels to condition their spectral weights on the specific phase of the physical process (e.g., distinguishing initial injection from late-stage saturation). During training, these sub-intervals are treated as independent samples, effectively increasing the batch size by a factor of N_{splits} while proportionally reducing the temporal dimension of each tensor. During inference, the model queries the required time windows sequentially or in parallel, and the outputs are reintegrated (through concatenation or super-resolution) to reconstruct the full history. Figure 1 illustrates our approach. In practical terms, considering a baseline STU approach with a single sample ($N_s = 1$), 4 snapshots ($N_t = 4$), and 1 input channel ($N_{c_{in}} = 1$), the input tensor size is $(1, 1, 4, N_x, N_y)$. In our proposed approach, splitting the temporal domain into two subdomains transforms this single trajectory into two independent samples ($N_s = 2$), each with half the snapshots ($N_t^c = 2$). With the addition of the time coordinate channel, the input tensor becomes $(2, 2, 2, N_x, N_y)$. With that, the discrete convolution domain (last three axis) would contain only half of the components on the STU approach.

3 NUMERICAL RESULTS

We apply our strategy to known benchmarks in the reservoir simulation community. The SPE10 (Christie & Blunt, 2001) is a set of two problems initially proposed to test upscaling techniques in



(a) Spatio-temporal unified (STU) approach.



(b) Time-splitting with coordinate injection approach.

Figure 1: Scheme illustrating an example where the proposed scheme can be used. When comparing the STU approach with our proposed scheme, the number of snapshots is divided by the number of splits in the temporal domain $\mathcal{T}$. This approach leads to the addition of temporal coordinates as an input channel and a proportional multiplication of the number of samples. Reintegration can be done through superresolution or concatenation.

reservoir simulation. In summary, in this study we focus on SPE10 Model 1 as a proof of concept, with the goal of later expanding this method for SPE10 Model 2, a substantially larger problem. SPE10 Model 1 is a 2D problem containing one gas injection well, one production well and the reservoir is fully saturated with oil. Details about governing equations, the numerical solver, the SPE10 benchmark, and the machine learning workflow are addressed in Appendix B. We vary the gas injection rate of the injector well in a uniform distribution of $\mathcal{U}(0.24, 0.37)$ Mscf/day for a total of 100 numerical simulations, where 80 trajectories are used for training and 20 for validation. Each trajectory is assessed for 8000 days, with $\Delta t = 10$ days, leading to 800 snapshots. With that, we use TFNO (Kossaifi et al., 2024), a tensorized version of FNO, to predict the gas saturation field, the oil production rate, and cumulative oil production for the whole time horizon. We test splitting the 800 snapshots into chunks of 400, 200, 100 and 50 snapshots, leading to a total of 2, 4, 8 and 16 splits in time. Models are trained on a NVIDIA H100 with 94 GB of VRAM. In terms of model generalization quality, we assess the relative L^2 error ($\eta(t) = ||\mathbf{Y} - \hat{\mathbf{Y}}||_2/||\mathbf{Y}||_2$, where $\mathbf{Y}$ is the ground truth and $\hat{\mathbf{Y}}$ is the prediction using the surrogate model) for all 20 test samples. Given that we assess the relative error in time, we average the errors across all the other dimensions (that is, all samples, channels and spatial dimensions). We extend our analysis to primitive variables (gas saturation fields) and quantities of interest (oil production rates and totals). To measure efficiency, we monitor GPU power and memory usage during model training in Weights & Biases (Biewald, 2020).

Fig. 2 shows the model's generalization error. The left figure shows the relative error in time. The relative error in time is below 10^{-2} for almost all time steps across all cases. We notice that η increases for the STU approach when t approaches the end of the simulation interval. When splitting the temporal domain, we observe the same behavior at the artificial temporal boundaries of each sub-window ($\mathcal{T}^c$). For the individual channels, this phenomenon is more pronounced in the gas saturation channel. Nonetheless, these localized errors do not significantly propagate to the primary Quantities of Interest (QoI); the oil production rate and cumulative oil production show strong resilience to these split-point fluctuations, and all tested configurations remain within good

agreement with the ground truth, as seen in Fig. 3. Further analysis in model generalization is done in Appendix C.

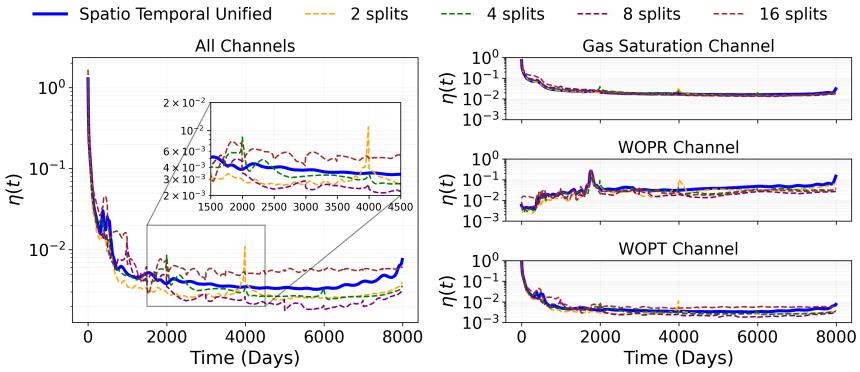


Figure 2: Relative error in time for all splits averaged across all 20 samples in the test set.

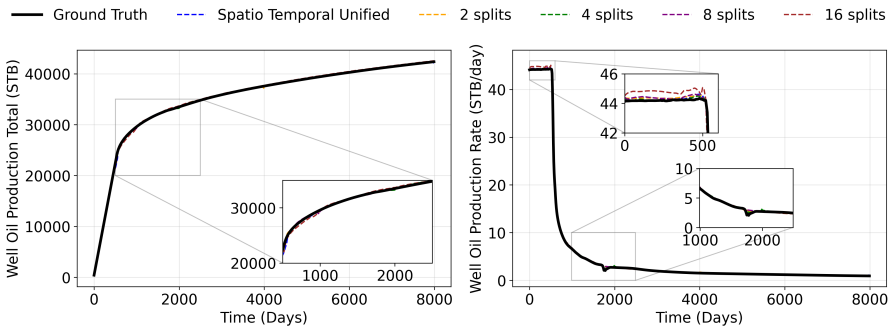


Figure 3: Quantities of Interest predicted by the FNO model for all considered splits.

Now, assessing the relation between the splitting strategy and hardware utilization, as measured by GPU usage on Weights & Biases, we observe nonlinear resource consumption behavior, as depicted in Fig. 4. We plot the results for wall time, given that all jobs were executed sequentially using a Hydra (Yadan, 2019) sweep. We note that increasing the split density generally correlates with sustained higher GPU power usage, likely because smaller tensor dimensions reduce I/O time and increase computing time. This correlation is reinforced when we look at the time spent accessing memory, with the 16-split curve being the most steady. Conversely, the GPU memory allocated decreases as the temporal window size is reduced, though this relationship is not strictly monotonic. While configurations for 2, 4, and 8 splits show moderate variations affected by allocator fragmentation and cache management, the 16-split configuration achieves a substantial reduction in peak memory usage, reaching approximately 5.3% of the total GPU memory, which is 7 times less than the STU approach. This collapse in memory footprint suggests a critical threshold where sub-window activations fit entirely within primary hardware cache bins, effectively bypassing the need for larger global memory reservations.

4 Conclusions

In this study, we assessed the use of a time-splitting strategy that uses time as a parameter on FNO surrogate modeling through the use of explicit time coordinate injection as input channels. We tested our approach on a challenging benchmark proposed by the petroleum engineering community. Our results indicate that while finer splitting introduces localized boundary errors, it provides a promising venue for executing large-scale reservoir simulations on memory-constrained hardware without compromising the accuracy of integrated production metrics. Most notably, we observed a 7-fold improvement in memory efficiency for the 16 splits case versus the STU model. This demonstrates that the marginal increase in boundary error is a justifiable trade-off for a drastic reduction

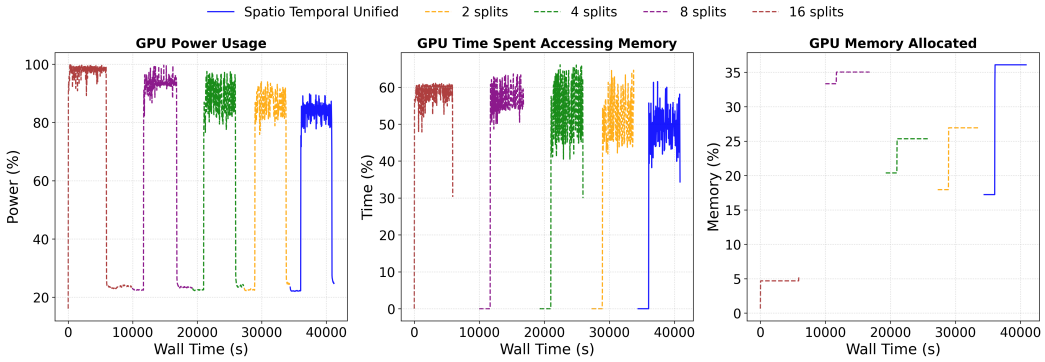


Figure 4: Efficiency metrics for the tested cases. Wall times in seconds counting from the start of the code's execution. Five runs are executed sequentially due to Hydra's parameter sweep run.

in computational footprint, being a promising strategy for deploying complex SciML models on resource-limited infrastructures.

Conflict of Interest Statement

The authors have no conflicts of interest to declare that are relevant to the content of this article.

Acknowledgments

This study was partially funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brasil (CAPES)—Finance Code 001. It is also partially supported by CNPq, the Brazilian Petroleum Agency, and ExxonMobil.

References

D. Badawi and E. Gildin. Neural operator-based proxy for reservoir simulations considering varying well settings, locations, and permeability fields. *Computers & Geosciences*, 196:105826, February 2025. ISSN 0098-3004. doi: 10.1016/j.cageo.2024.105826.

L. Biewald. Experiment tracking with weights and biases, 2020. URL `https://www.wandb.com/`. Software available from wandb.com.

A. O. Bocoum and M. R. Rasaei. Multi-objective optimization of WAG injection using machine learning and data-driven Proxy models. *Applied Energy*, 349:121593, November 2023. ISSN 0306-2619. doi: 10.1016/j.apenergy.2023.121593.

M. A. Christie and M. J. Blunt. Tenth SPE Comparative Solution Project: A Comparison of Upscaling Techniques. *SPE Reservoir Evaluation & Engineering*, 4(04):308–317, August 2001. ISSN 1930-0212. doi: 10.2118/72469-pa.

K. H. Coats, L. K. Thomas, and R. G. Pierson. Compositional and Black Oil Reservoir Simulation. *SPE Reservoir Evaluation & Engineering*, 1(04):372–379, August 1998. ISSN 1930-0212. doi: 10.2118/50990-pa.

Ø. Fevang, K. Singh, and C. H. Whitson. Guidelines for Choosing Compositional and Black-Oil Models for Volatile Oil and Gas-Condensate Reservoirs. In *SPE Annual Technical Conference and Exhibition*, 00ATCE. SPE, October 2000. doi: 10.2118/63087-ms.

T. J. Grady, R. Khan, M. Louboutin, Z. Yin, P. A. Witte, R. Chandra, R. J. Hewett, and F. J. Herrmann. Model-parallel Fourier neural operators as learned surrogates for large-scale parametric PDEs. *Computers & Geosciences*, 178:105402, September 2023. ISSN 0098-3004. doi: 10.1016/j.cageo.2023.105402.

J. Kossaifi, N. Kovachki, Z. Li, D. Pitt, M. Liu-Schiaffini, R. J. George, B. Bonev, K. Azizzadenesheli, J. Berner, V. Duruisseaux, and A. Anandkumar. A Library for Learning Neural Operators. *arXiv*, 2024. doi: 10.48550/arxiv.2412.10354.

N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: learning maps between function spaces with applications to PDEs. *The Journal of Machine Learning Research*, 24(1), January 2023. ISSN 1532-4435.

W. Lee, T. Kim, and H. Park. Fourier Neural Operators for Non-Markovian Processes:Approximation Theorems and Experiments. *arXiv*, 2025. doi: 10.48550/arxiv.2507.17887.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. *arXiv*, 2020. doi: 10.48550/arxiv.2010.08895.

J. Liu, H. Jing, and H. Pan. New Fast Simulation of 4D (x, y, z, t) CO2 EOR by Fourier Neural Operator Based Deep Learning Method. In *Society of Petroleum Engineers - SPE Reservoir Simulation Conference, RSC 2023*, 2023. ISBN 9781613998717. doi: 10.2118/212236-MS.

R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*, 2018.

M. McCabe, P. Harrington, S. Subramanian, and J. Brown. Towards Stability of Autoregressive Neural Operators. *arXiv*, 2023. doi: 10.48550/arxiv.2306.10619.

B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12346 LNCS, pp. 405–421, 2020. ISBN 9783030584511. doi: 10.1007/978-3-030-58452-8_24.

R. Ohana, M. McCabe, L. Meyer, R. Morel, F. J. Agocs, M. Beneitez, M. Berger, B. Burkhart, K. Burns, S. B. Dalziel, D. B. Fielding, D. Fortunato, J. A. Goldberg, K. Hirashima, Y.-F. Jiang, R. R. Kerswell, S. Maddu, J. Miller, P. Mukhopadhyay, S. S. Nixon, J. Shen, R. Watteaux, B. Régaldo-Saint Blancard, F. Rozet, L. H. Parker, M. Cranmer, and S. Ho. The Well: a Large-Scale Collection of Diverse Physics Simulations for Machine Learning. *arXiv*, 2025. doi: 10.48550/arXiv.2412.00568.

A. F. Rasmussen, T. H. Sandve, K. Bao, A. Lauser, J. Hove, B. Skaflestad, R. Klöfkorn, M. Blatt, A. B. Rustad, O. Sævareid, K. A. Lie, and A. Thune. The Open Porous Media Flow reservoir simulator. *Computers & Mathematics with Applications*, 81:159–185, January 2021. ISSN 0898-1221. doi: 10.1016/j.camwa.2020.05.014.

A. Samnioti and V. Gaganis. Applications of Machine Learning in Subsurface Reservoir Simulation—A Review—Part I. *Energies*, 16(16):6079, 2023. ISSN 1996-1073. doi: 10.3390/en16166079.

M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, volume 2020-December, 2020.

G. Wen, Z. Li, K. Azizzadenesheli, A. Anandkumar, and S. M. Benson. U-FNO — An enhanced Fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163, 2022. ISSN 03091708. doi: 10.1016/j.advwatres.2022.104180.

O. Yadan. Hydra - A framework for elegantly configuring complex applications. Github, 2019. URL `https://github.com/facebookresearch/hydra`.

Z. Ye, C. S. Zhang, and W. Wang. Recurrent Neural Operators: Stable Long-Term PDE Prediction. *arXiv*, 2025. doi: 10.48550/arXiv.2505.20721.

A TIME HANDLING STRATEGIES IN FOURIER NEURAL OPERATORS

In this appendix, we detail the mathematical formulations of the three distinct time-handling strategies for Fourier Neural Operators: the standard spatio-temporal formulation, the autoregressive approach, and the proposed time-splitting scheme. Common to all strategies is the general Fourier Neural Operator architecture (Kovachki et al., 2023), denoted as G_θ. The operator is a composition of a lifting layer P, a sequence of L iterative hidden layers, and a projection layer Q. An illustration of the standard FNO architecture can be seen in Fig. A1. Formally, for an input a, the network output is given by:

$$G_\theta(a) = (Q \circ \mathcal{L}_L \circ \cdots \circ \mathcal{L}_1 \circ P)(a) \tag{A1}$$

where P maps the input to the first hidden state $\mathbf{v}_0$, and Q projects the final hidden state $\mathbf{v}_L$ back to the target domain. The core of the operator lies in the iterative updates $\mathcal{L}_l$, which map $\mathbf{v}_{l-1} \to \mathbf{v}_l$. The update rule for a hidden layer l is defined as:

$$\mathbf{v}_l(\mathbf{x}, t) = \sigma\left(\mathbf{W}\mathbf{v}_{l-1}(\mathbf{x}, t) + (\mathcal{K}\mathbf{v}_{l-1})(\mathbf{x}, t)\right), \tag{A2}$$

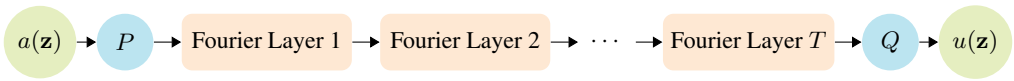
where $\mathcal{K}$ represents the integral kernel operator. For FNO (Li et al., 2020), the integral kernel operator is defined as

$$(\mathcal{K}\mathbf{v}_l)(\mathbf{x}, t) = \int_{\mathcal{D}} \kappa(\mathbf{x} - \hat{\mathbf{x}}, t - \tau)\mathbf{v}_l(\tau)d\tau\,, \tag{A3}$$

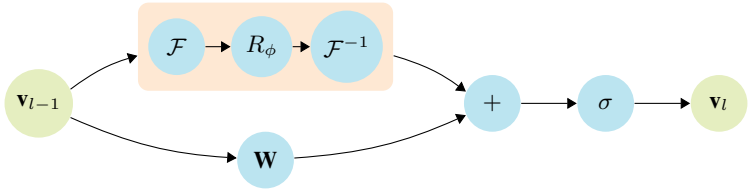
where $\mathbf{x}$ and $\hat{\mathbf{x}}$ are the spatial coordinates, t and τ define the temporal coordinate and $\mathcal{D}$ is the convolution domain. In practice, computing the integral in eq. A3 directly is computationally prohibitive. By invoking the Convolution Theorem, we can evaluate this operation efficiently in the spectral domain. The kernel integration is thus implemented as a multiplication by a learnable weight tensor R_ϕ in the Fourier space:

$$(\mathcal{K}\mathbf{v}_l)(\mathbf{z}) = \mathcal{F}^{-1}\left(R_\phi \cdot \mathcal{F}(\mathbf{v}_l)(\mathbf{k})\right)(\mathbf{z}), \tag{A4}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier Transform and its inverse over the domain $\mathcal{D}$, respectively, $\mathbf{k}$ represents the frequency modes and $\mathbf{z} \in \mathcal{D}$ denotes the generalized coordinate vector (e.g., $\mathbf{z} = (\mathbf{x}, t)$ or $\mathbf{z} = \mathbf{x}$), ensuring the formulation applies consistently across the distinct strategies defined below. The tensor R_ϕ contains the learnable complex weights that parameterize the kernel κ directly in the frequency domain. That said, the distinction between how FNO handles time lies in the definition of the domain $\mathcal{D}$, the definition of the convolution variables, and the integration limits.



(a) FNO layer structure



(b) Fourier layer l

Figure A1: Fourier Neural Operator architecture. Figure (a) shows the overall structure, in which the input is first lifted to a higher dimension through operator P, then passed through Fourier layers, and lastly projected back into the output space dimension through operation Q. Figure (b) shows the structure of a Fourier layer, which comprises of the application of the FFT ($\mathcal{F}$), a linear transformation R_ϕ and the reverse FFT ($\mathcal{F}^{-1}$), then the result is summed with a local linear transformation $\mathbf{W}$, and lastly a non-linear activation function σ is applied, generating the layer output.

A.1 STANDARD SPATIO-TEMPORAL UNIFIED FORMULATION

In the standard formulation (often referred to as 4D FNO for 3D time-dependent problems), time is treated as an intrinsic dimension of the convolution domain (Ye et al., 2025). In this approach,

convolution is performed on a domain $D = \Omega \times \mathcal{T} \subset \mathbb{R}^3 \times \mathbb{R}^+$, where Ω is the spatial domain and $\mathcal{T} = [0, T]$ is the temporal domain, where T is full temporal horizon. The kernel integration is performed globally over both space and time, such that

$$(\mathcal{K}\mathbf{v}_l)(\mathbf{x}, t) = \int_0^T \int_\Omega \kappa(\mathbf{x} - \hat{\mathbf{x}}, t - \tau)\mathbf{v}_l(\hat{\mathbf{x}}, \tau)\, d\hat{\mathbf{x}}\, d\tau. \tag{A5}$$

In this scheme, the learned kernel κ captures global spatio-temporal correlations. Consequently, the spectral operations must be performed over the joint space-time domain. The implementation of the kernel operator extends eq. A5 to include the temporal frequency modes ω:

$$(\mathcal{K}\mathbf{v}_l)(\mathbf{x}, t) = \mathcal{F}^{-1}\left(R_\phi(\mathbf{k}, \omega) \cdot \mathcal{F}(\mathbf{v}_l)(\mathbf{k}, \omega)\right)(\mathbf{x}, t), \tag{A6}$$

where ω represents the frequency mode corresponding to the time dimension t. This highlights the computational cost of this approach: the Fourier Transform $\mathcal{F}$ must be computed as a $(d+1)$-dimensional operation (e.g., 4D FFT for 3D time-dependent problems), processing the entire space-time history simultaneously.

A.2 Autoregressive Formulation

The autoregressive approach treats the FNO as a state-transition function (or time-stepper) that evolves the system from t to $t + \Delta t$. Time is handled iteratively via a feedback loop rather than within the spectral kernel (McCabe et al., 2023). Here, the convolution domain is $\mathcal{D} = \Omega \subset \mathbb{R}^3$ and the convoluted coordinates are $\mathbf{x}$ and $\hat{\mathbf{x}}$. Time t is effectively a channel index or an external iteration variable and this process usually requires reshaping the input tensor (a practical perspective can be seen in at the official website of (Ohana et al., 2025)). The kernel integration is strictly spatial, being

$$(\mathcal{K}\mathbf{v}_l)(\mathbf{x}) = \int_\Omega \kappa(\mathbf{x} - \hat{\mathbf{x}})\mathbf{v}_l(\hat{\mathbf{x}})\, d\hat{\mathbf{x}}. \tag{A7}$$

In this approach, we can formally describe the temporal evolution of the output state $\mathbf{u}$ to be recovered by the composition of the operator G_θ:

$$\mathbf{u}(\mathbf{x}, t + k\Delta t) = \underbrace{G_\theta \circ G_\theta \circ \cdots \circ G_\theta}_{k \text{ times}}(\mathbf{u}(\mathbf{x}, t))\,, \tag{A8}$$

where Δt is the time step size used in the autoregressive FNO roll-out. This iterative structure highlights the fundamental trade-off of the autoregressive strategy. While the kernel integration in eq. A7 is computationally lighter per step (requiring only d-dimensional spatial FFTs compared to the $(d+1)$-dimensional space-time FFTs of the unified formulation), the recursive composition in eq. A8 introduces the challenge of error accumulation. Since the input to G_θ at step k is the predicted state from step $k-1$ (which inherently contains approximation error), these errors can compound over long time horizons, potentially leading to divergent trajectories. This contrasts with the global approach in eq. A6, where the error is distributed holistically over the temporal domain $\mathcal{T}$.

A.3 Time-Splitting with Coordinate Injection

In this proposed method, we introduce a hybrid strategy that partitions the global spatio-temporal domain $\mathcal{D}$ into a sequence of sub-intervals $\mathcal{T}_i$. This allows for high-dimensional spectral convolutions (as in the unified approach) but restricted to manageable temporal windows $[\tau_i, \tau_{i+1}]$, significantly reducing the memory footprint. The domain for the i-th sub-problem is defined as $\mathcal{D}_i = \Omega \times [\tau_i, \tau_{i+1}] \subset \mathbb{R}^3 \times \mathbb{R}^+$. A critical challenge with pure time-splitting is the loss of temporal context: standard FNO kernels are shift-invariant, meaning G_θ would process the interval $[0, \Delta t]$ exactly the same as $[T - \Delta t, T]$ if the input feature values were identical. To resolve this ambiguity and break the shift-invariance with respect to absolute time, we inject the absolute temporal coordinate directly into the feature space. The input to the first layer is reformulated as a vector-valued function $\tilde{\mathbf{v}}_{in} : \mathcal{D}_i \to \mathbb{R}^{d_v+1}$, defined component-wise as:

$$\tilde{\mathbf{v}}_{in}(\mathbf{x}, t)_j = \begin{cases} \mathbf{v}_{in}(\mathbf{x}, t)_j & \text{if } 1 \le j \le d_v \\ \lambda(t) & \text{if } j = d_v + 1 \end{cases} \tag{A9}$$

where $\lambda(t)$ is the coordinate embedding (or raw value) of the absolute time, and d_v is the number of elements in the vector $\mathbf{v}_{in}(\mathbf{x}, t)$. This formulation explicitly distinguishes the coordinate injection, which increases the feature dimension, from time-handling strategies that concatenate outputs along the temporal dimension. The kernel integration is then performed over the local sub-window, utilizing this enriched state:

$$(\mathcal{K}\mathbf{v}_l)(\mathbf{x}, t) = \int_{\tau_i}^{\tau_{i+1}} \int_{\Omega} \kappa(\mathbf{x} - \hat{\mathbf{x}}, t - \tau)\tilde{\mathbf{v}}_l(\hat{\mathbf{x}}, \tau)\, d\hat{\mathbf{x}}\, d\tau. \tag{A10}$$

By injecting absolute time, the operator G_θ becomes effectively non-stationary, allowing the learned kernels to condition their spectral weights on the specific phase of the physical process.

While this formulation preserves the ability to model local temporal derivatives via $(d + 1)$-dimensional FFTs (unlike the spatial-only autoregressive kernel), it implicitly imposes a finite temporal receptive field. Correlations spanning beyond the window size $|\tau_{i+1} - \tau_i|$ are severed. However, the coordinate injection defined in eq. A9 acts as a global reference, partially mitigating this by providing the network with a reference to its global position in the time horizon. This strategy aims to balance between the computational tractability of autoregressive steps and the rich expressivity of global spectral convolutions seen in the spatio-temporal unified approach.

This approach has limitations, however. First, by decomposing $\mathcal{T}$ into smaller subdomains $\mathcal{T}_i^c$, we are effectively truncating the temporal receptive field of the spectral operator. Mathematically, the global convolution integral, which naturally aggregates information across the entire domain $\mathcal{T}$, is replaced by a localized integration strictly bounded within the subdomain. Figure A2 illustrates this phenomenon, where the orange region is the contribution to $(\mathcal{K}\mathbf{v}_l)(\mathbf{x}, t)$ in the temporal domain for the Spatio-Temporal Uniform approach and our proposed method. The blue region covers the discarded interactions by using truncated temporal subdomains. This localization implicitly forces the learned kernel $\kappa(t - \tau)$ to zero for any lag $|\Delta t|$ outside the bounds of the subdomain size, thereby severing any long-range temporal correlations that span across subdomains. The effects of truncation are physics-dependent, and we investigate this phenomenon in our test cases. Second, this truncation can introduce ambiguity in systems with time-varying inputs or strong path dependence. If no data augmentation strategy is adopted, we could have different outputs in $\mathcal{T}_2^c$ for two samples with identical input channels on $\mathcal{T}_2^c$ if they were different in $\mathcal{T}_1^c$. Finally, by sectioning $\mathcal{T}$ into subdomains $\mathcal{T}^c$, we add artificial boundaries into a continuous dimension, and it is known that FNO accuracy decreases near coordinate boundaries (Wen et al., 2022). We explore two strategies to mitigate this issue in Appendix C.

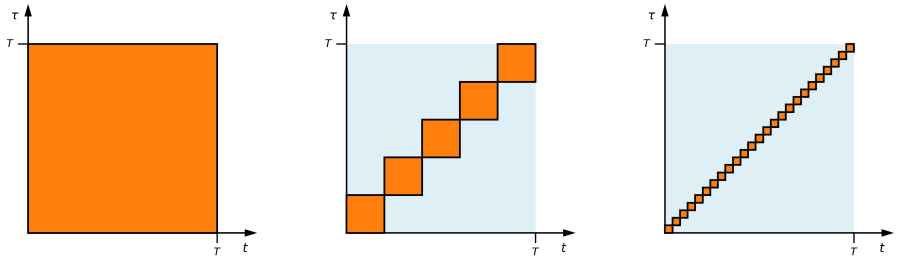


Figure A2: Illustration of the temporal domain information being convoluted. The orange region shows the retained information while the blue region represents information unseen by the convolution. The left figure represents the Spatio-Temporal Unified approach, center and right plots represent our proposed method, for different time-splitting windows.

B Computational Setup

B.1 Physical Model and Governing Equations

The mathematical framework governing multiphase fluid dynamics in porous media is derived from the fundamental conservation laws of mass, momentum, and energy, combined with thermodynamic

equilibrium constraints (Samnioti & Gaganis, 2023). In the context of reservoir simulation, modeling approaches are generally categorized into two primary families: black-oil models and compositional fluid models.

Black-oil formulations are designed to represent simplified phase behavior. They operate under the assumption that fluids flowing from the reservoir to surface facilities can be described as a binary mixture comprising stock tank oil and surface gas. Within this framework, phase behavior is quantified via PVT properties that fluctuate solely based on pressure and temperature, ignoring the specific chemical composition of the fluid. Conversely, compositional models track the detailed evolution of fluid composition throughout the spatial domain at every time step. This requires solving complex stability and flash calculations utilizing an Equation of State (EoS). While compositional models offer higher fidelity, they are computationally more demanding than their black-oil counterparts. For a comprehensive comparative analysis of these modeling strategies, readers are directed to Fevang et al. (2000); Coats et al. (1998).

For the purposes of this research, we employ the black-oil formulation to simulate multiphase flow. The mathematical structure of this model is grounded in mass conservation for each pseudo-component $\alpha \in \{w, o, g\}$, representing water, oil, and gas, respectively. This yields a system of coupled, non-linear partial differential equations:

$$\frac{\partial}{\partial t}\left(\phi_{\text{ref}} A_\alpha\right) + \nabla \cdot \mathbf{u}_\alpha + q_\alpha = 0 \ , \tag{B1}$$

where the first term denotes the rate of mass accumulation and the second term defines the mass flux. The accumulation terms, A_α, and their corresponding component velocities, $\mathbf{u}_\alpha$, are expressed as:

$$A_w = m_\phi b_w s_w \ , \qquad \mathbf{u}_w = b_w \mathbf{v}_w \ , \tag{B2}$$

$$A_o = m_\phi (b_o s_o + r_{og} b_g s_g) \ , \qquad \mathbf{u}_o = b_o \mathbf{v}_o + r_{og} b_g \mathbf{v}_g \ , \tag{B3}$$

$$A_g = m_\phi (b_g s_g + r_{go} b_o s_o) \ , \qquad \mathbf{u}_g = b_g \mathbf{v}_g + r_{go} b_o \mathbf{v}_o \ , \tag{B4}$$

Here, ϕ_{ref} represents the reference porosity, m_ϕ is a pressure-dependent multiplier, b_α denotes the formation volume factor (shrinkage/expansion), s_α is the phase saturation, and r_{go} and r_{og} represent the dissolved gas-oil ratio and the vaporized oil-gas ratio, respectively.

The velocities of the components, $\mathbf{u}_\alpha$, are intrinsically linked to the phase fluxes, $\mathbf{v}_\alpha$, which are governed by the multiphase generalization of Darcy's law:

$$\mathbf{v}_\alpha = -\lambda_\alpha \mathbf{K} \left(\nabla p_\alpha - \rho_\alpha \mathbf{g}\right) , \tag{B5}$$

where $\mathbf{K}$ is the absolute permeability tensor, λ_α represents phase mobility (defined as relative permeability normalized by viscosity), p_α is the phase pressure, ρ_α is the phase density, and $\mathbf{g}$ is the gravitational acceleration vector.

Beyond these governing PDEs, the system requires two auxiliary physical constraints. First, the saturation of all phases within the pore volume must sum to unity:

$$s_w + s_o + s_g = 1 \ . \tag{B6}$$

Second, the phase pressures are related via capillary pressure, p_c, which is modeled as a function of saturation:

$$p_{c,ow}(s_w) = p_o - p_w \ , \tag{B7}$$

$$p_{c,og}(s_g) = p_g - p_o \ . \tag{B8}$$

Numerical solutions for these black-oil equations are obtained using the OPM Flow simulator (Rasmussen et al., 2021). We utilize the 10th SPE Comparative Solution Project (SPE10) (Christie & Blunt, 2001) as our benchmark problem. Specifically, we focus on Model 1 of the SPE10 suite as a proof-of-concept. This model describes a two-phase (oil and gas) reservoir free of faults or dipping. The domain dimensions are $2500\text{ft} \times 25\text{ft} \times 50\text{ft}$, discretized into a $100 \times 1 \times 20$ grid. The reservoir is initially fully oil-saturated, with gas injected at a constant rate to drive production.

Figure B1 illustrates the permeability field, which is isotropic yet highly heterogeneous. Despite the low dimensionality of SPE10 Model 1, the permeability varies across 6 orders of magnitude. Operational parameters include a benchmark injection rate of 0.30 Mscf/day, a uniform porosity of 0.2, and a production well bottomhole pressure (BHP) maintained at 95 psia. Full parameter details for the SPE10 model are provided by Christie & Blunt (2001).

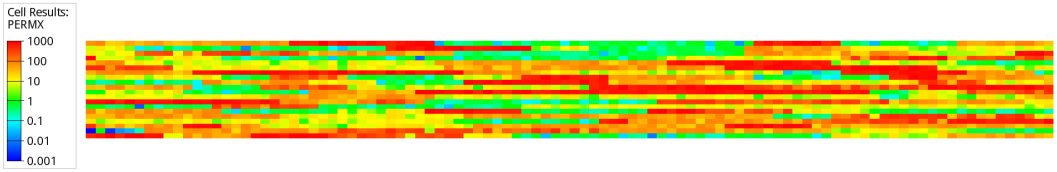


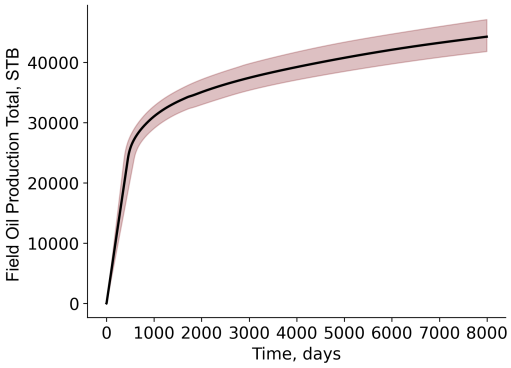
Figure B1: Correlated permeability field along the x-axis. Note that the logarithmic scale highlights a variation in permeability spanning 6 orders of magnitude. The vertical z-axis is exaggerated by a factor of 5 for visualization purposes.

B.2 Machine Learning Workflow

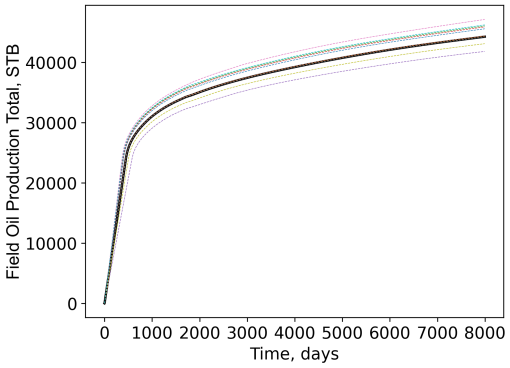
To construct the dataset for our machine learning framework, we conduct a suite of 100 simulations based on the SPE10 Model 1 setup. Variation is introduced through the gas injection rates, sampled as:

$$q_g = \mathcal{U}(0.23, 0.37)\,, \tag{B9}$$

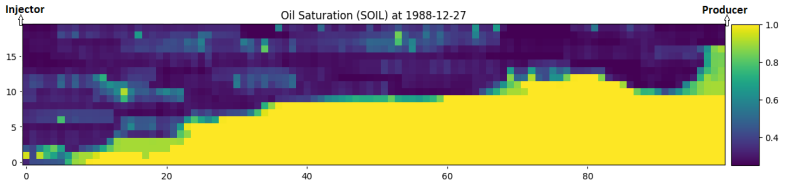
where $\mathcal{U}(a_1, a_2)$ denotes a uniform distribution. For each sampled q_g, a complete simulation is executed, yielding saturation and pressure fields alongside oil production curves. The envelope of cumulative oil production across all simulations, as well as selected individual curves, is depicted in Figure B2a.



(a) Envelope of total field oil production for 100 simulations.



(b) Sample of 10 field oil production curves.



(c) Oil saturation field at the final report step for the benchmark case (injection rate of 0.30 Mscf/day).

Figure B2: Visualization of numerical simulation outputs. The top-left panel displays the production envelope for the 100 simulations, while the top-right panel highlights a subset of 10 curves. In both plots, the black line represents the standard SPE10 Model 1 solution. The bottom panel illustrates the oil saturation distribution at the final time step for the benchmark case (0.30 Mscf/day). Injection rates were sampled uniformly between 0.23 Mscf/day and 0.37 Mscf/day.

Throughout each simulation, OPM Flow generates two primary data types: grid fields and summary values. Grid fields represent spatio-temporal variables defined for every mesh cell at each report step, such as pressure and component saturations. An example of such a field—specifically oil saturation—is presented in Figure B2c. Conversely, summary values are scalar quantities computed via post-processing at every solver time step, such as well production rates and bottomhole pressures. For example, integrating the oil saturation (a grid field) using the well model logic within OPM Flow yields the scalar production values shown in Figure B2b.

Since OPM Flow utilizes binary formats compatible with Schlumberger's ECLIPSE simulator, the industry standard for reservoir engineering (Rasmussen et al., 2021), we employ the 'res2df' package [1] developed by Statoil/Equinor to extract and organize both grid and scalar data.

Both grid-based and post-processed variables are essential for reservoir analysis and for training proxy models. However, their distinct data structures pose a challenge for unified machine learning ingestion. Geological inputs like porosity and permeability are static spatial maps, whereas operational parameters such as injection rates and BHP variations are dynamic temporal scalars. To integrate these disparate formats into a single tensor representation, we adopt the bit mask strategy introduced by Badawi & Gildin (2025). In this approach, well locations within the spatial grid are assigned their corresponding scalar values in a zero-filled tensor. Consequently, temporal scalars are converted into spatio-temporal tensors where the well's grid position carries the non-zero scalar value.

This transformation process is reversible and is also applied to extract scalar predictions from the surrogate model (e.g., predicting the oil production rate at a specific time). By identifying the grid cells associated with a production well, we average the predicted tensor values at those locations to recover the scalar quantity. This workflow is illustrated in Figure B3. Once the input tensors are assembled, the models are trained. We apply normalization to the data using the Unit Gaussian normalizer from the neuralop library (Kovachki et al., 2023; Kossaifi et al., 2024).

B.3 Model Architecture and Training Protocol

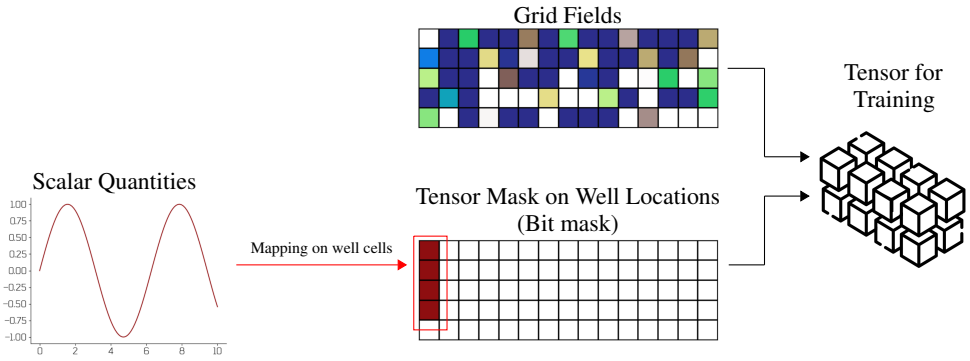
The surrogate model employed in this work is a Tensor Factorized Neural Operator (TFNO) (Kossaifi et al., 2024). It is a more efficient version of the standard FNO, which uses a Tucker tensor factorization to compress the spectral weights. After initial ablation studies for this problem, we selected a rank fraction of 0.05, which balances computational efficiency with the representational capacity required to capture fine-scale flow features. The network is constructed with a lifting layer that projects the input features into a higher-dimensional latent space with a hidden width of 64 channels. This is followed by 4 spectral convolution layers. Within each layer, the Fourier transform is computed, and a subset of the frequency modes is retained to perform the spectral convolution. Given the 2D nature of the SPE10 Model 1 ($100 \times 1 \times 20$) and the temporal dimension, we explicitly define the number of Fourier modes as $[32, 32, 1, 32]$ for the time, x, y, and z axes, respectively. The non-linear activation function used between these spectral layers is the Rectified Linear Unit (ReLU).

la The model is trained on a dataset comprising both static and dynamic fields. The input tensor consists of three channels: the static permeability field (PERMX), the time variable (YEARS), and the well gas injection rate (WGIR). The model is tasked with predicting three distinct output channels simultaneously: the dynamic gas saturation grid (SGAS), and two scalar well outputs injected into the grid via the bit-mask method—the Well Oil Production Rate (WOPR) and the Well Oil Production Total (WOPT). To facilitate stable convergence, we employ a Unit Gaussian Normalizer, which centers and scales the data pixel-wise for both inputs and outputs, effectively standardizing the distribution of the heterogeneous permeability and saturation fields.

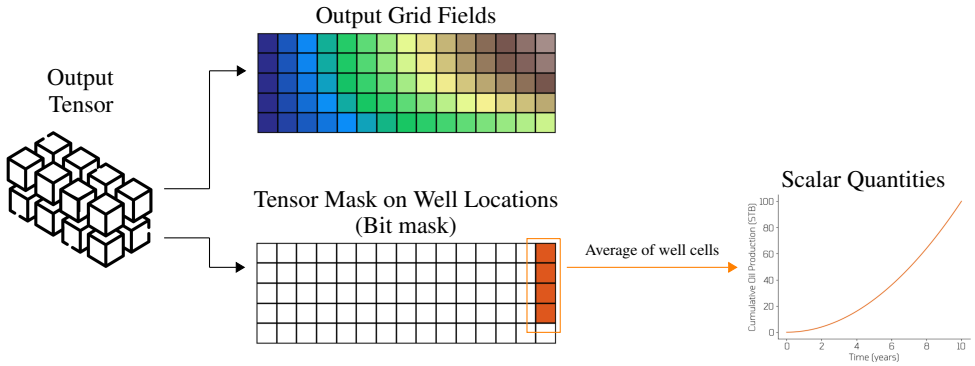
The optimization process is driven by the Adam optimizer, initialized with a learning rate of 8×10^{-4} and a weight decay of 1×10^{-4} to regularize the weights and prevent overfitting. The objective function minimized during training is the relative L^2 loss (LpLoss with $p = 2$), calculated over the entire domain. To dynamically adjust the optimization trajectory, we employ a 'ReduceLROnPlateau' learning rate scheduler. This scheduler monitors the validation loss and reduces the learning rate by a factor of 0.5 if the loss fails to improve for a patience period of 10 epochs. A cooldown period of 15 epochs is also enforced to prevent excessive reduction. The dataset is randomly partitioned into a training set (80%) and a test set (20%), and the model is trained for a total of 300 epochs with a batch size of 10 samples, ensuring sufficient exposure to the varying injection rate scenarios sampled in the dataset.

This setup is preserved for all tested cases and we perform all training, inference and visualization steps on a High Performance Computing (HPC) environment. We use Hydra (Yadan, 2019) to perform parameter sweep and Weights and Biases (Biewald, 2020) to track our experiments. Taking

[1] `https://equinor.github.io/res2df/index.html`



(a) Adding scalar curves to the surrogate model.



(b) Extracting scalar curves from the surrogate model.

Figure B3: Schematic of scalar quantity management within the proposed architecture. Panel (a) demonstrates the transformation of scalar values into tensors using bit masks at well locations. Panel (b) depicts the reverse process: converting output tensors back into scalar quantities by averaging cell values at the well locations for each time step.

advantage of Hydra's structure, we override certain items on the configuration files (such as padding, data augmentation and number of time splits) to avoid writing over the scripts themselves for each experiment. With that, training runs are executed sequentially under that specific context.

C Ablation Studies on Boundary Effects

Results shown on Fig. 2 indicate that splitting the time domain $\mathcal{T}$ leads to the addition of artificial boundaries. It is known that FNO struggles on coordinate boundaries (Wen et al., 2022). Although this effect does not introduce significant errors in our models, we investigate two strategies to mitigate it in this appendix. We analyze the impact of boundary mitigation techniques (spectral padding and data augmentation) and different reintegration strategies (concatenation vs. super-resolution). Our objective is to determine whether the computational overhead of these additional mechanisms yields proportional performance gains compared to the baseline Coordinate Injection method.

C.1 PADDING

It is well-documented that Fourier Neural Operators can suffer from spectral leakage and Gibbs phenomena at domain boundaries due to the implicit periodicity assumption of the FFT (Wen et al., 2022; Lee et al., 2025). A common mitigation strategy is zero-padding the domain prior to the spectral convolution. In our time-splitting approach, partitioning the temporal domain $\mathcal{T}$ introduces artificial boundaries at every split point t_i. We hypothesized that zero-padding the temporal dimension by 10% and 20% might smooth the error spikes observed at these interfaces.

First, we investigate the evolution of the mean relative error across all 20 test samples. That is, we average the errors across samples (N_s), channels ($N_{c_{out}}$), and spatial coordinates (N_x, N_y, N_z), and plot their evolution across the 800 snapshots (reintegration being done by concatenating the subdomains). As illustrated in the comparison plots described in the Figs. C1, padding alone does not effectively mitigate the errors introduced by the artificial boundaries in the split models. We observe the same behavior as in Fig. 2, meaning that the use of zero-padding did not address the issue we initially thought could be solved. We further quantify how padding affects each of our models by computing the Frobenius norm of the entire tensor for each test sample. Results are available in the boxplots in Fig. C2. We observe that the relation between zero-padding and our strategy is nonlinear. For the spatio-temporal unified approach and the 8-split case, adding padding shifted the distributions to larger errors; that is, as we increased the padding percentage, we observed a proportional increase in the relative error distribution for the 20 samples in the test set. For the cases of 4 and 16 splits, adding more padding shifted the distribution towards smaller errors in test-set predictions. In the case of 2 splits, the distribution had no practical change whether padding was added or not. In most cases, the distribution did not change the interquantile distance much. Although these results are counter-intuitive, they suggest that the proposed Coordinate Injection mechanism (Eq. A9) is the dominant factor in stabilizing the operator. By explicitly providing the absolute time coordinate, the network successfully learns to handle the "start" and "end" dynamics of each sub-window without requiring artificial spectral smoothing. In terms of Quantities of Interest, Fig. C3 shows how the assessment of the QoIs using padding matches the core results described in Fig. 3 with minor differences.

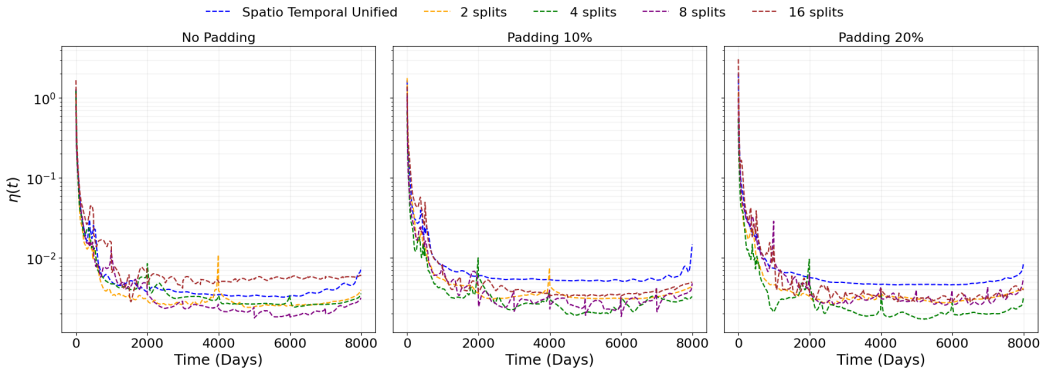


Figure C1: All channels relative error for all considered splits averaged across all 20 samples in the test set. The figure on the left shows the original results with no padding (also displayed in Fig. 2), the central graph displays the relative errors when using 10% padding, and the figure on the right shows them using 20% padding.

C.2 DATA AUGMENTATION VIA SLIDING WINDOWS

To address boundary discontinuities, we implemented a data augmentation strategy that used overlapping (sliding) time windows during training. Instead of non-overlapping partitions $[t_i, t_{i+1}]$, we generated training samples with overlapping temporal domains, effectively treating the artificial boundary points of one sample as internal points of another. This approach ensures that the model treats the artificial boundary time steps as internal points across different training samples, thereby theoretically smoothing the transition. Figure C4 illustrates how our data augmentation strategy is proposed. Following the example in the illustration, for a given sample containing 8 snapshots,

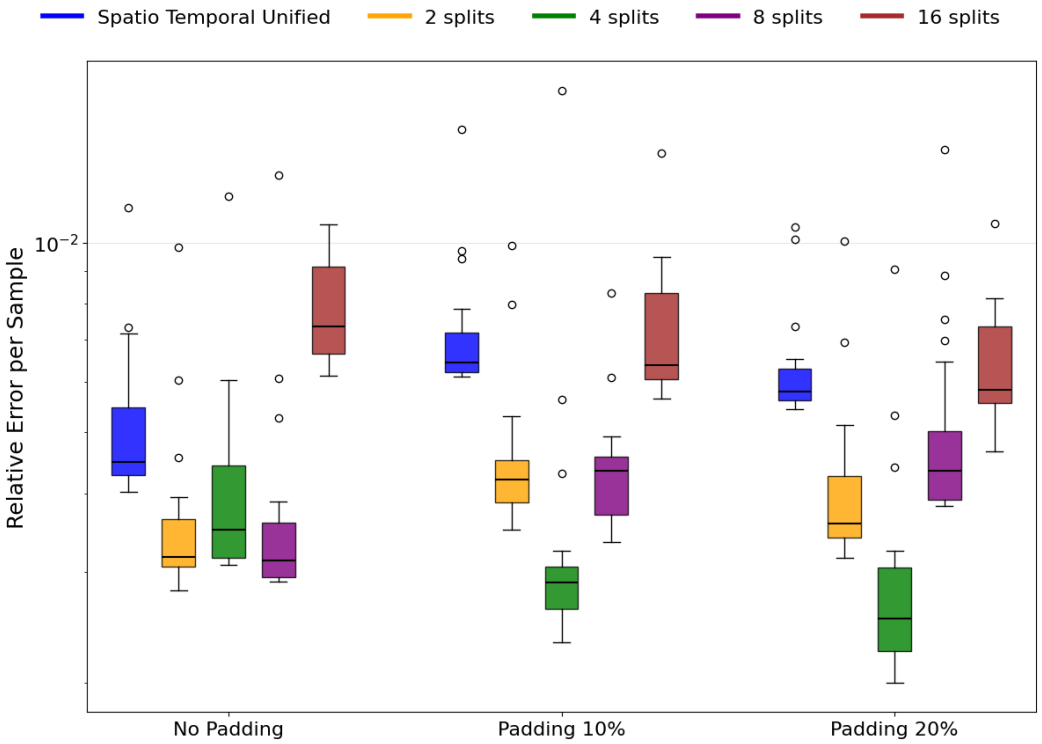


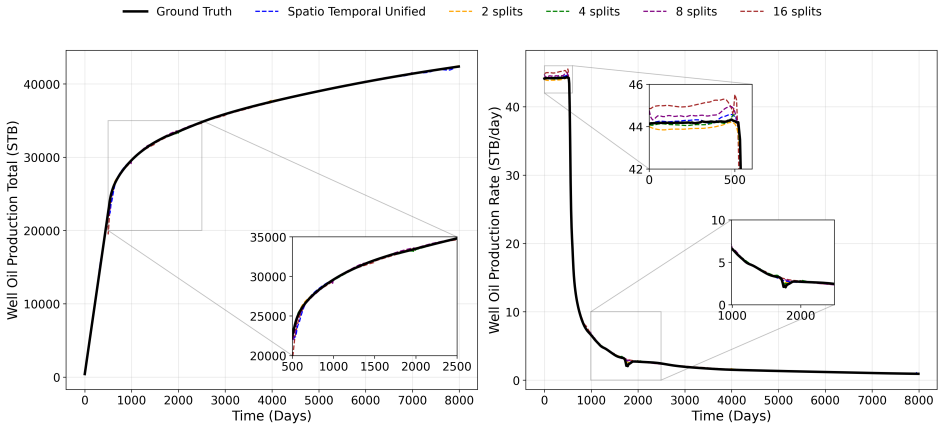
Figure C2: Box plots for the relative error of the 20 test set samples, averaged across all channels and all time steps. The plot compares the relative errors of the models without padding (on the left) with models that apply 10% (center) and 20% (right) padding.

using our time-splitting strategy for 2 splits, 2 independent samples are generated, each having 4 snapshots. New Sample 1 now maps the discrete dynamics of $t = \{t_1, t_2, t_3, t_4\}$, and Sample 2 is responsible for describing the discrete dynamics for $t = \{t_5, t_6, t_7, t_8\}$. In the original proposed scheme, $t = \{t_4, t_5\}$ would be artificial borders added to the spectral convolution operator. For this scheme, we add a new training sample, Sample 3, that maps the dynamics $t = \{t_3, t_4, t_5, t_6\}$. This way, the FNO model can hopefully learn the dynamics for $t = \{t_4, t_5\}$ without being affected by the artificial borders.

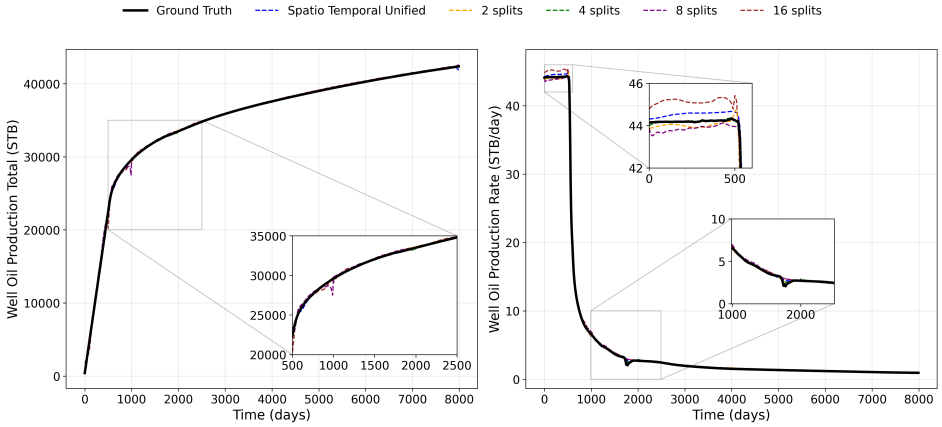
We perform an ablation study using data augmentation and compare the results with those obtained in Section 3. The results shown are in the Fig. C5. We notice that, visually, the error cusps at the artificial boundary points are slightly smoothed in the augmented models compared to the use of padding, as shown in Fig. C1. However, this strategy does not, per se, mitigate this issue. By analyzing the overall error for the test samples in Fig. C6, we notice the same behavior seen with zero-padding. We show the results for all splits tested. Given that this strategy uses a sliding window for data augmentation and the spatio-temporal uniform approach uses the entire window in the spectral convolution, these strategies are not compatible. For 2 and 16 splits, gains are marginal, whereas for 4 and 8 splits, results worsen with data augmentation. While data augmentation theoretically provides more examples of boundary transitions, it increases the training time linearly with the number of overlaps. Given that the baseline method (non-overlapping splits) already achieves engineering-grade accuracy on QoIs (seen in Fig. 3), we conclude that the standard partitioning strategy strikes the optimal balance between training efficiency and accuracy.

C.3 Super-resolution

In Section 2, we note that predictions can be reintegrated in two ways. Core results of this study show that the temporal domain $\mathcal{T}$ gets decomposed into several subdomains $\mathcal{T}^c$ and the model can predict the output fields for $\mathcal{T}^c$ that can be glued together to compose $\mathcal{T}$. In this section, we take advantage of the resolution-invariance property of FNO to one-shot predict outputs on $\mathcal{T}$ and show



(a) QoIs when adding 10% zero-padding.



(b) QoIs when adding 20% zero-padding.

Figure C3: Oil production total and rates computed for models using zero-padding.

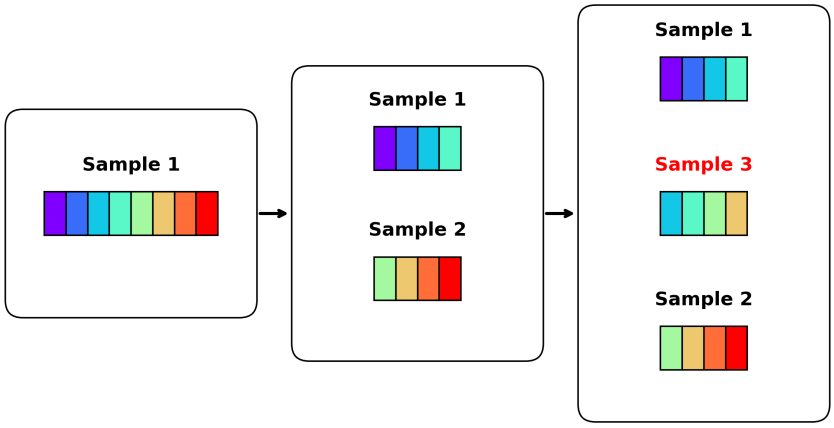


Figure C4: Illustration of the proposed data augmentation scheme. In this scheme, each block represents one snapshot in the problem dynamics. The first stage represents the full dynamics of one sample. The second stage comprises the time-splitting strategy, generating two new independent samples. The third stage illustrates how a third sample can be originated from the combination of the previous samples.

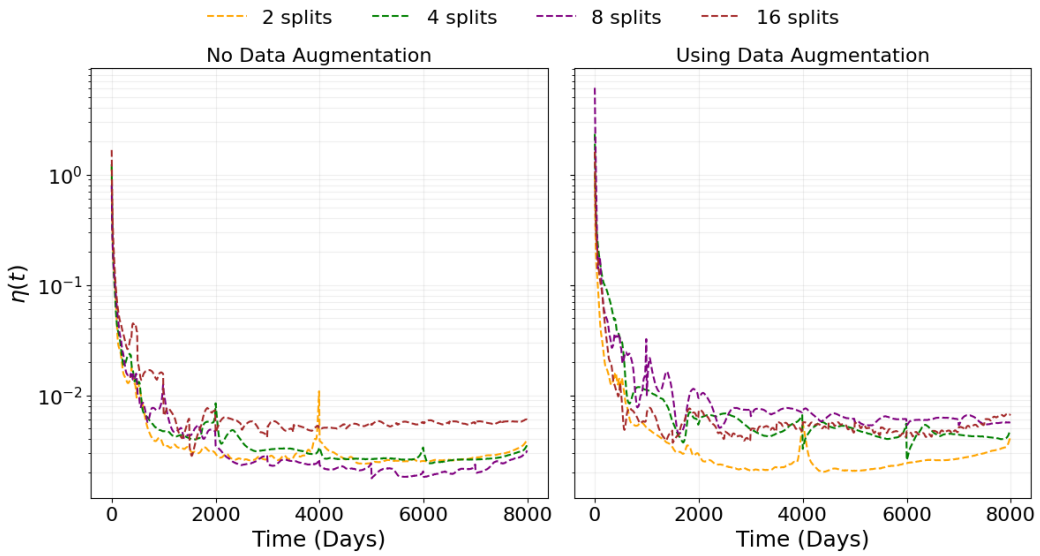


Figure C5: All channels relative error for all considered splits averaged across all 20 samples in the test set. The figure on the left shows the original results with no data augmentation (also displayed in Fig. 2), while the figure on the right shows the relative error when data augmentation is applied.

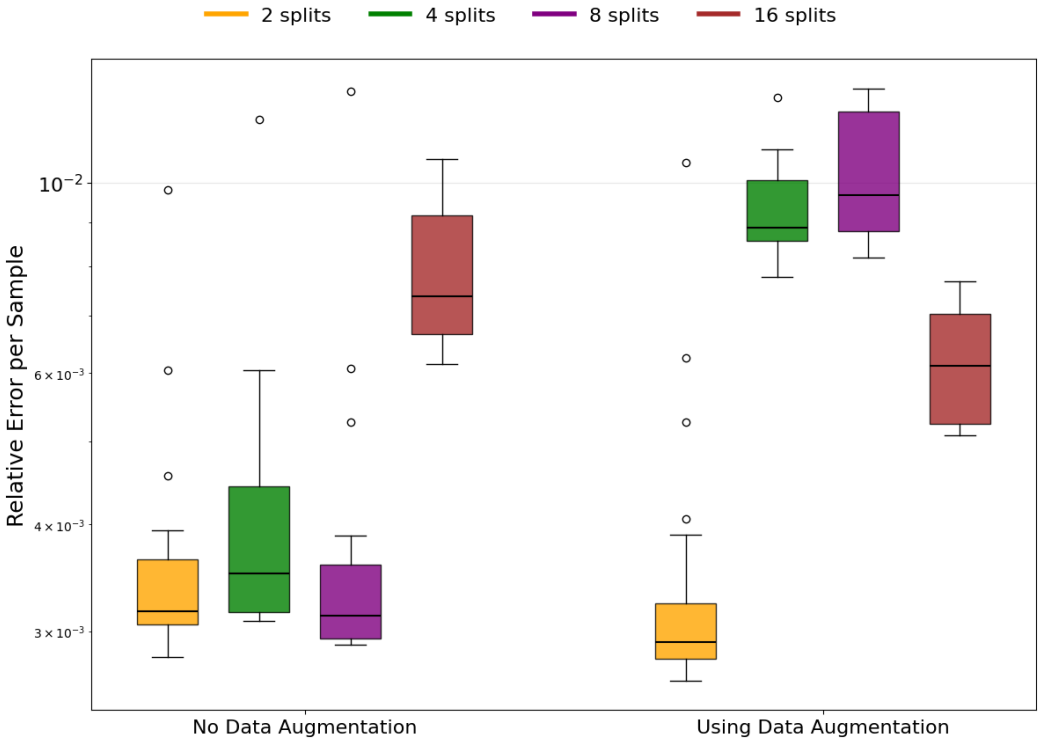


Figure C6: Box plots for the relative error of the 20 test set samples, averaged across all channels and all time steps. The plot compares the relative errors of the models without (left) and with (right) data augmentation.

another way to reintegrate the predictions over the total time horizon expected. That is, from one sample in $\mathcal{T}^c$, the input channels provide information for extrapolating to the temporal domain $\mathcal{T}$. Figure C7 shows that super-resolution in this case results in high errors that degrade periodically

for all test cases. The same phenomenon can be observed in Fig. C9, where the models attempt to predict the oil production curves for a single test sample. This confirms that the learned operator G_θ is specialized to the spectral frequency distribution of the sub-window size. Although the curves are not ideal for engineering purposes, it is interesting to see that the model with a substantially reduced input tensor can exhibit trends in long-term predictions for these quantities of interest. It is important to note that both the concatenation and super-resolution strategies yield the same results as the spatio-temporal unified approach. This is due to two facts: when using concatenation as our reintegration strategy, the predictions cover the entire temporal domain $\mathcal{T}$, whereas in super-resolution, both the input and the output share the high-dimensional information for N_t. That is, in both cases, the STU approach does not require any reintegration and yields exactly the same results. Another interesting result is that the number of modes is the same for all tested splits, as described in Section B.3. Considering that for each assessed number of splits we get a smaller number of snapshots in time for each split, the same number of Fourier modes can be enough to capture the dynamics in one case and not in others. We notice that for 8 and 16 splits not only the overall relative error between predictions and ground truths for the test set is slightly smaller in Fig. C8, but we notice how these curves have a better match to the pattern seen in the WOPT plot in Fig. C9.

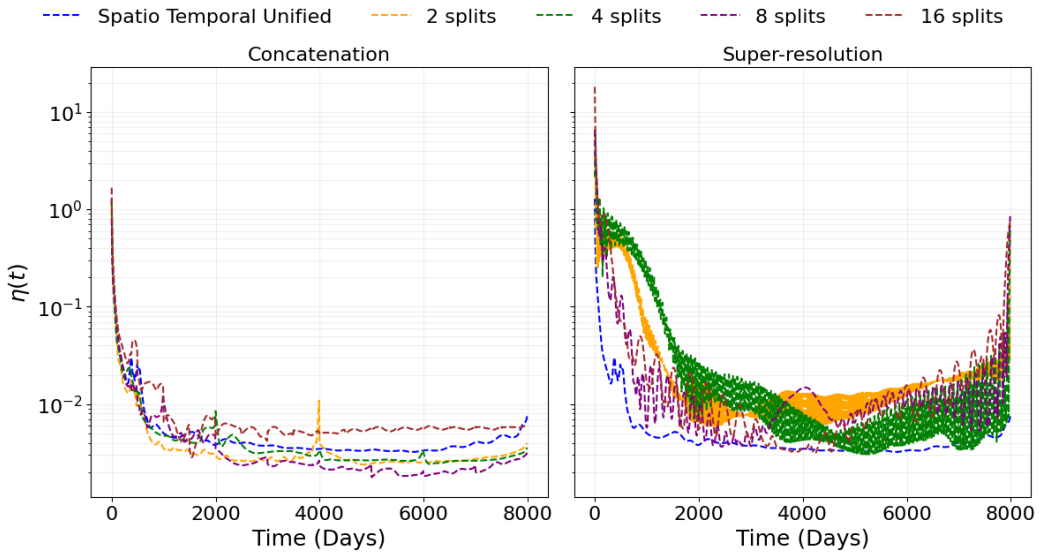


Figure C7: All channels relative error for all considered splits averaged across all 20 samples in the test set. The figure on the left shows the original results with no data augmentation (also displayed in Fig. 2), while the figure on the right shows the relative error when super-resolution is attempted.

C.4 Grid Fields

As the core results of the paper, depicted in Figure 2 and 3, we show how temporal manipulation can affect quantities of interest in oil simulation and how the model can generalize (in terms of the average of the relative error between the ground truth and the prediction in time). In this section, we also analyze how the spatial error is affected by our time-splitting scheme. Figures C10, C11 and C12 show the gas saturation field for a test sample at $t = 490$ days, $t = 3990$ days and $t = 7990$ days, respectively. That is, the 49th, the 399th, and the 799th snapshots of our prediction. These limits are chosen because $t = 490$ days corresponds to an early stage of the dynamics and is the last snapshot for the first split in the 16-split case in our model. For $t = 3990$ days, the solution is exactly half of the total time horizon and near an artificial boundary that affects all splits tested. For $t = 7990$ days, it is near the end of the total time available for the physics predicted. The pointwise difference map shows that errors are concentrated primarily at the sharp saturation fronts (shock waves). Importantly, the plume's structural integrity is preserved, and the saturation front location matches the ground truth in all cases. For all cases, the difference between predictions and ground truth is negligible. These small differences are also evident in the histograms, where, despite different absolute error variances, most errors are below 10^{-1}. These low errors confirm that the

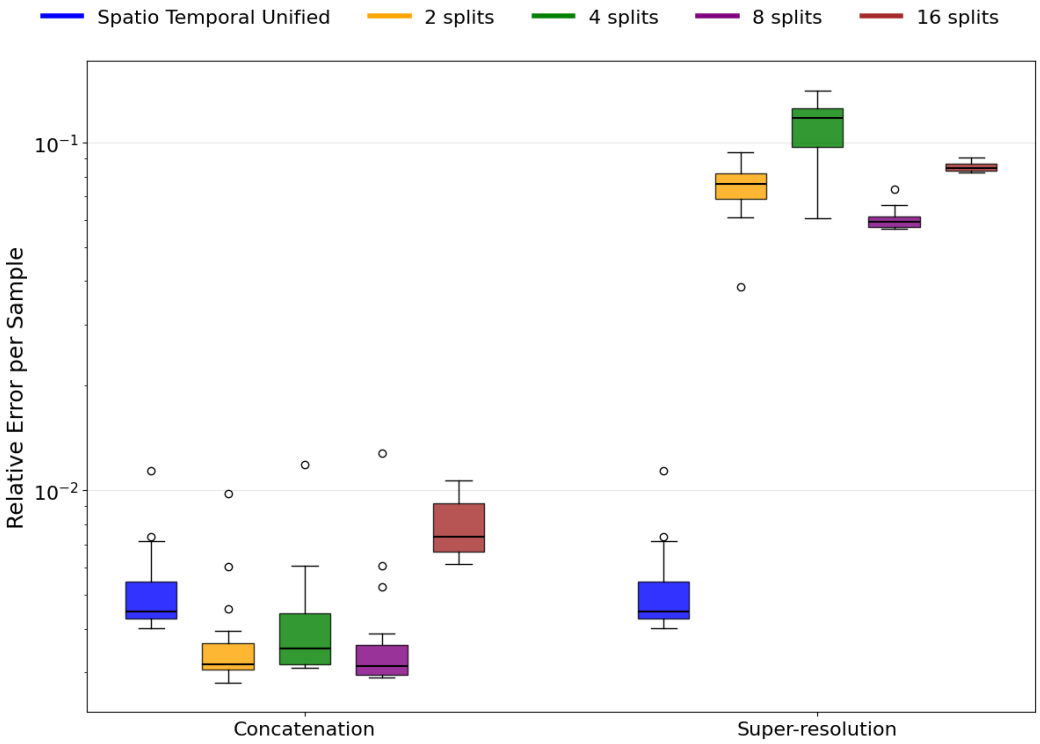


Figure C8: Box plots for the relative error of the 20 test set samples, averaged across all channels and all time steps. The plot compares the relative errors of the models to forecast within the training time-resolution (left) to super-resolution forecast (right).

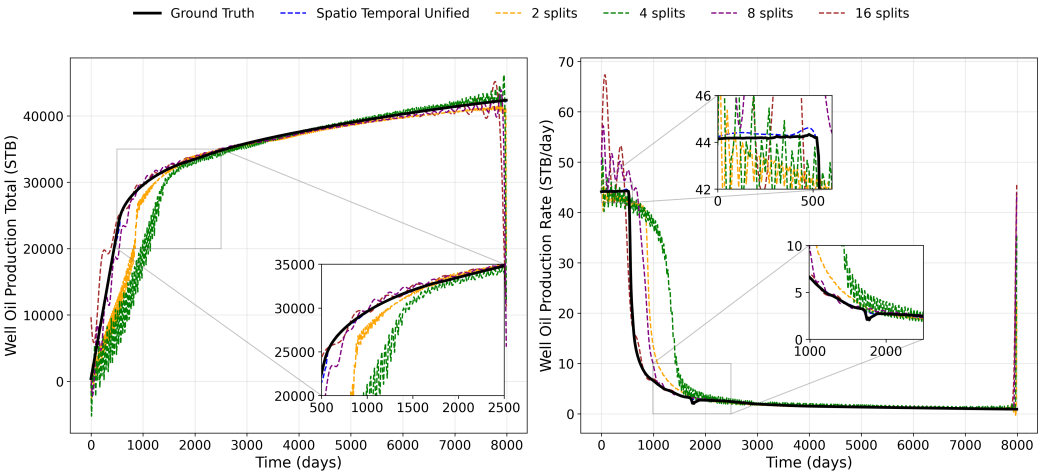
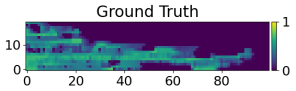
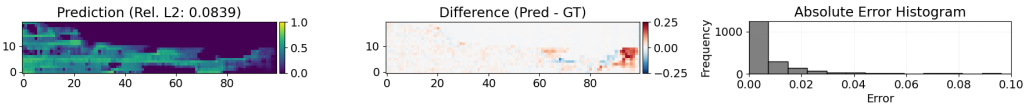


Figure C9: Quantities of Interest throughout time of operation for super-resolution forecast on a test sample. Left plot comprises the cumulative oil production volume over time while right plot covers the oil production rate. The black curves correspond to the ground truth the models wish to ascertain, and the colored dashed lines correspond to the model's predictions, for each time-splitting scheme.

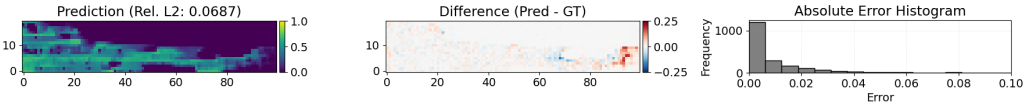
Time-Splitting FNO successfully captures the underlying physics of multiphase flow, with errors resembling numerical dispersion in traditional solvers.



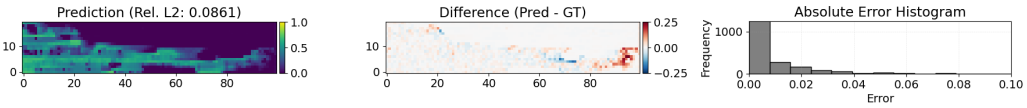
(a) Ground truth.



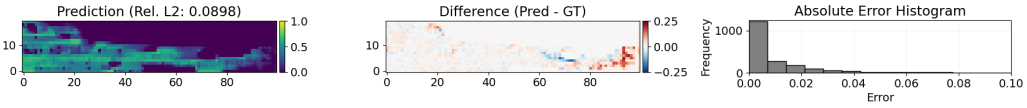
(b) Spatio-Temporal Unified approach.



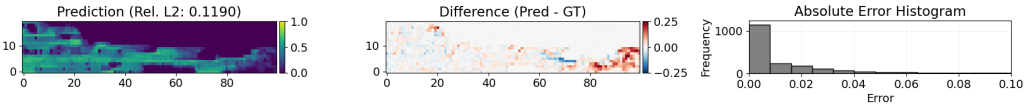
(c) 2 splits.



(d) 4 splits.

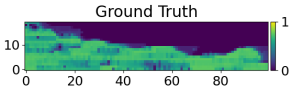


(e) 8 splits.

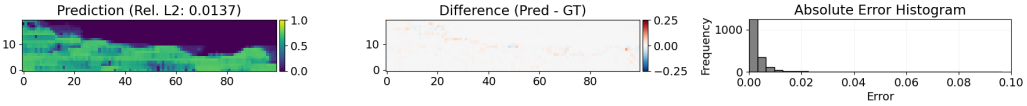


(f) 16 splits.

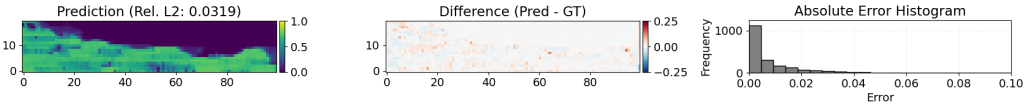
Figure C10: Prediction of a test set sample at $t = 490$ days using different time splitting strategies.



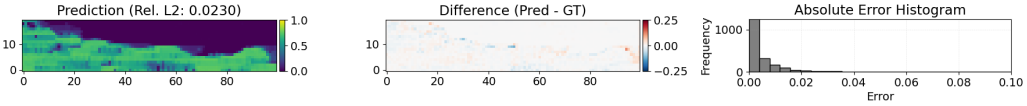
(a) Ground truth.



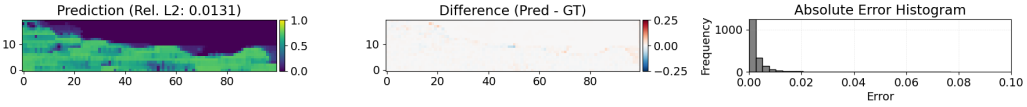
(b) Spatio-Temporal Unified approach.



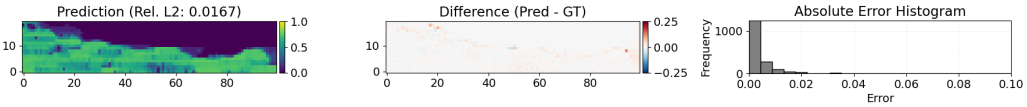
(c) 2 splits.



(d) 4 splits.

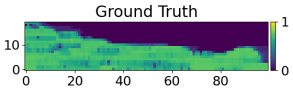


(e) 8 splits.

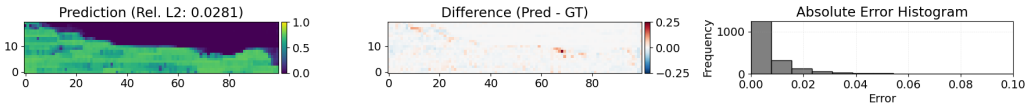


(f) 16 splits.

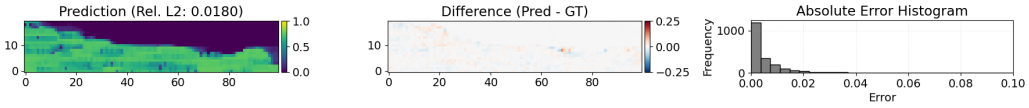
Figure C11: Prediction of a test set sample at $t = 3990$ days using different time splitting strategies.



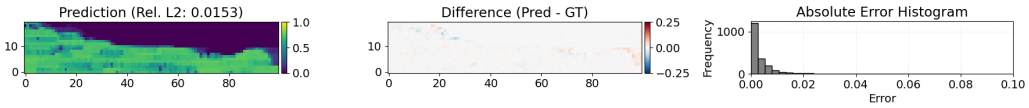
(a) Ground truth.



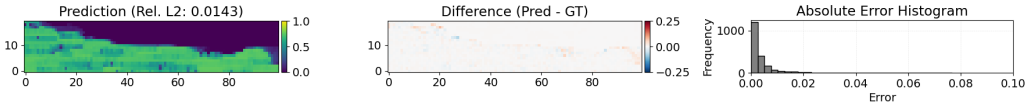
(b) Spatio-Temporal Unified approach.



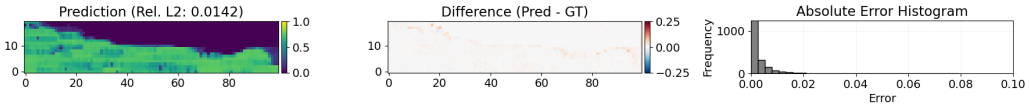
(c) 2 splits.



(d) 4 splits.



(e) 8 splits.



(f) 16 splits.

Figure C12: Prediction of a test set sample at $t = 7990$ days using different time splitting strategies.