DCNO: IMPROVING OPERATOR LEARNING THROUGH MULTI-STREAM DYNAMIC CONNECTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformer-based neural operators offer strong representational capacity for solving partial differential equations (PDEs). However, preserving early latent physical information while allowing representations to evolve remains a challenge as operator depth increases. To address this problem, we propose the Dynamic Connection Neural Operator (DCNO), which assigns information persistence and representation evolution to distinct but interacting latent streams. Specifically, DCNO maintains a slowly evolving operator-memory stream alongside more freely evolving working streams, with each operator layer dynamically reading from all streams. This design keeps early latent information accessible while allowing working representations to evolve through progressive transformations. Extensive experiments across diverse PDE benchmarks demonstrate that DCNO consistently improves predictive performance, while depth-focused analyses further reveal more effective utilization of increased operator depth.

1 INTRODUCTION

Partial differential equations (PDEs) are fundamental tools for modeling physical processes such as fluid flow, heat transfer, and elastic deformation (Evans, 2022). Classical numerical methods, such as finite element methods (FEM), solve such equations by discretizing the governing equations over the computational domain (Brenner & Scott, 2008). However, repeatedly performing high-fidelity simulations under varying boundary conditions or geometries can be computationally expensive, especially in large-scale problems with high-dimensional parameter spaces (Azizzadenesheli et al., 2024). In contrast, neural operators provide a data-driven alternative by learning mappings between infinite-dimensional function spaces to approximate solution operators for families of PDEs (Lu et al., 2019; Li et al., 2020). Once trained, neural operators can predict solutions for new inputs through a single forward pass, providing efficient surrogate models for repeated scientific simulations (Kovachki et al., 2023).

As the underlying physical dependencies become more complex, neural operators require greater representational capacity to model them effectively. Transformer-based neural operators have therefore emerged as a popular choice, as attention can flexibly capture long-range, input-dependent interactions across discretized physical domains (Vaswani et al., 2017; Cao, 2021). Representative examples include OFormer (Li et al., 2022), which combines self- and cross-attention to learn solution operators with flexible input and query sampling; GNOT (Hao et al., 2023), which introduces heterogeneous normalized attention and geometric gating to accommodate multiple input functions and irregular meshes; and Transolver (Wu et al., 2024), which models physical correlations through attention over learned physical slices. These architectures typically rely on residual connections (He et al., 2016a) to propagate latent states across successive attention and feed-forward transformations, enabling progressive refinement through stacked operator blocks.

Despite these advances, a key challenge remains as Transformer-based neural operators grow deeper: latent representations should continue to evolve while retaining effective access to problem-related information. On the one hand, successive operator blocks need to transform their latent states to integrate spatial interactions and construct increasingly informative solution features. On the other hand, geometry, coefficient fields, forcing terms, and initial or boundary conditions define the physical problem being solved (Evans, 2022; Kovachki et al., 2023), and later operator computations should remain effectively conditioned on this information. Under standard residual propagation, the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

same evolving latent state both carries information forward and accumulates successive operator updates. Although residual connections provide a direct identity path for information propagation (He et al., 2016b; Team et al., 2026), successive operator updates still accumulate in the same evolving latent representation. As depth increases, such repeated modification may reorganize the condition-related structure encoded in the latent state. This suggests a potential tension between information persistence and representation evolution.

Figure 1 provides a preliminary representation-level view of this persistence–evolution tension on the Heat2D dataset. We examine how latent representations evolve relative to the early-stage encoded physical conditions across depth by GNOT (Hao et al., 2023), a representative Transformer-based neural operator. As network depth increases, the inter-sample structure of GNOT's latent representations becomes progressively less aligned with the relational structure induced by the encoded physical conditions, decreasing from approximately 0.38 at block 0 to about 0.08 at block 7. This decline provides preliminary empirical support for the persistence–evolution tension and motivates our central question: *Can a neural operator maintain persistent access to problem-related information while still allowing sufficiently flexible representation evolution across depth?*

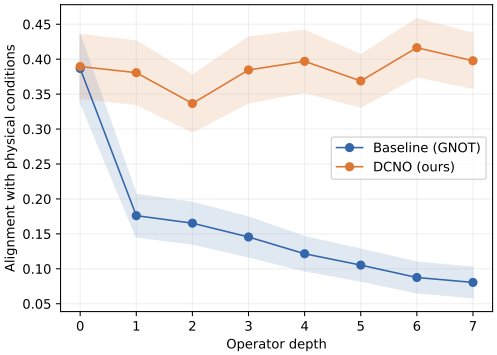


Figure 1: Physical-condition alignment across operator depth on Heat2D, measured against the same encoded-condition reference for both models. DCNO maintains more stable alignment as depth increases.

To address this question, we propose the *Dynamic Connection Neural Operator* (DCNO), a multi-stream connection architecture for deep neural operators. To maintain an accessible latent reference alongside evolving solution representations, DCNO initializes one operator-memory stream and several working streams from the same encoded state. The memory stream is protected from incoming cross-stream mixing and receives scaled updates to encourage gradual adaptation, while the working streams mix and transform more freely to construct new solution features. Each operator layer dynamically reads from these streams, allowing its use of the memory to change without requiring comparable changes to the memory itself. As shown in Figure 1, DCNO maintains more stable alignment with the encoded physical conditions across depth than the baseline GNOT. Experiments on diverse PDE benchmarks show consistent improvements in prediction accuracy over a broad range of baselines. Further depth-scaling experiments demonstrate greater predictive benefits from additional operator layers, supporting the effectiveness of this propagation design. Our main contributions are summarized as follows:

- **Persistence–evolution perspective for deep neural operators.** We identify a tension between information persistence and representation evolution across depth, and introduce a representation-level diagnostic to characterize this behavior.
- **Dynamic Connection Neural Operator.** We propose DCNO, a multi-stream architecture that maintains a slowly evolving operator-memory stream alongside flexible working streams, keeping persistent information accessible throughout operator computation.
- **Empirical validation across tasks and depths.** Experiments on seven PDE benchmarks show consistent improvements over baselines, while depth and representation analyses demonstrate stronger information persistence and more effective use of deeper layers.

2 Related Work

2.1 Neural Operators for PDEs

Neural operators learn mappings between function spaces to approximate PDE solution operators. DeepONet (Lu et al., 2019) uses branch and trunk networks to represent input functions and query locations, while MIONet (Jin et al., 2022) extends this formulation to multiple inputs. The Fourier

Neural Operator (FNO) (Li et al., 2020) parameterizes integral kernels in Fourier space, with subsequent developments targeting multiscale modeling through U-NO (Rahman et al., 2022), U-FNO (Wen et al., 2022), and multiwavelet-based operators (Gupta et al., 2021). F-FNO (Tran et al., 2021) factorizes spectral transformations, while Latent Spectral Models (LSM) (Wu et al., 2023) perform spectral computation in learned latent spaces. To handle irregular geometries and non-uniform discretizations, GNO (Kovachki et al., 2023) uses graph kernels, Geo-FNO (Li et al., 2023b) learns geometric deformations, and NUNO (Liu et al., 2023) employs domain partitioning. GINO (Li et al., 2023c) combines graph and Fourier operators for local and global modeling.

Attention-based neural operators learn input-dependent interactions over discretized domains. Galerkin Transformer (Cao, 2021) introduces efficient linear attention, and OFormer (Li et al., 2022) uses self- and cross-attention to process input observations and output queries separately. FactFormer (Li et al., 2023a) explores factorized attention for multiscale modeling and efficiency. GNOT (Hao et al., 2023) combines heterogeneous normalized attention with geometric gating to accommodate multiple inputs and irregular meshes, while ONO (Xiao et al., 2023) develops orthogonal attention for operator approximation. LNO (Wang & Wang, 2024) uses Physics-Cross-Attention to map between geometric and latent spaces. Transolver (Wu et al., 2024) aggregates mesh points into learnable slices and applies Physics-Attention to the resulting tokens. Transolver++ (Luo et al., 2025) extends this approach to million-scale geometries through adaptive slice mechanisms and parallel computation. Xia & Aviles-Rivero (2026) introduce geometry memory injection to reinforce geometric conditioning at intermediate depths.

2.2 Connection Architectures for Deep Networks

Connection architectures govern representation propagation and reuse across depth. ResNet (He et al., 2016a) introduces additive shortcuts, while pre-activation residual units emphasize direct identity paths for forward and backward propagation (He et al., 2016b). Highway Networks (Srivastava et al., 2015) use learned gates to regulate transformed and carried features. Subsequent approaches control residual updates through learnable scaling: ReZero (Bachlechner et al., 2021) uses zero-initialized scalar gates, while LayerScale (Touvron et al., 2021) introduces per-channel scaling initialized at small values. DeepNorm (Wang et al., 2024) combines modified residual connections with depth-dependent initialization to stabilize deep Transformer training. DenseNet (Huang et al., 2017) enables direct reuse of earlier features through concatenation, while Dual Path Networks (Chen et al., 2017) combine residual and dense pathways.

Learned aggregation further generalizes interactions across layers and streams. Dynamic Linear Combination of Layers (DLCL) (Wang et al., 2019) and DenseFormer (Pagliardini et al., 2024) learn weighted combinations of preceding layer or block outputs, whereas Attention Residuals (Team et al., 2026) uses content-dependent attention over earlier outputs, with a block-wise variant to reduce memory overhead. Hyper-Connections (HC) (Zhu et al., 2025) extend residual propagation to multiple interacting streams through learned input aggregation, stream mixing, and output redistribution. Their dynamic variant conditions connection weights on the current hidden state. Manifold-Constrained Hyper-Connections (mHC) (Xie et al., 2025) impose doubly stochastic constraints on cross-stream residual mappings by Sinkhorn–Knopp iterations to improve propagation stability.

3 Dynamic Connection Neural Operator

We introduce the *Dynamic Connection Neural Operator* (DCNO), a multi-stream connection architecture that supports information persistence and representation evolution through distinct latent streams. DCNO maintains a slowly evolving memory stream alongside more freely evolving working streams, while allowing each operator block to adaptively access information across them. This design modifies how information is propagated across depth without changing the computation within the underlying neural-operator blocks. Figure 2 illustrates the overall architecture.

3.1 Dynamic Connection with Operator Memory

Multi-stream latent state. Let $h_0 \in \mathbb{R}^D$ denote the encoded representation at a query location, where D is the latent dimension. Inspired by Hyper-Connections (Zhu et al., 2025), DCNO replicates h_0 into $r \geq 2$ latent streams, forming $H_0 = \mathbf{1}_r h_0^\top \in \mathbb{R}^{r \times D}$, where $\mathbf{1}_r \in \mathbb{R}^r$ is the all-ones

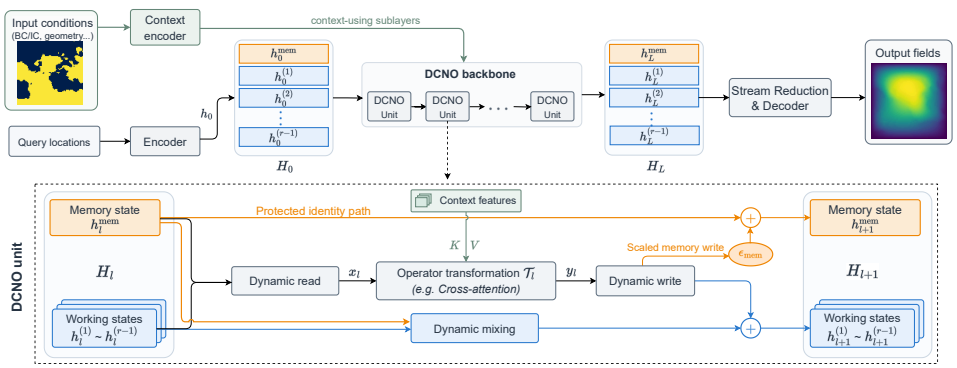


Figure 2: **Overall architecture of DCNO.** DCNO maintains a protected, slowly updated memory stream alongside dynamically evolving working streams, keeping persistent information accessible while allowing flexible representation transformation across depth.

vector. We index the DCNO units by $l = 0, \dots, L-1$, with each unit mapping H_l to H_{l+1}. At unit l, the latent state is $H_l = \left[h_l^{\text{mem}}, h_l^{(1)}, \dots, h_l^{(r-1)}\right]^\top \in \mathbb{R}^{r \times D}$, where $h_l^{\text{mem}} \in \mathbb{R}^D$ is the memory state and $h_l^{(1)}, \dots, h_l^{(r-1)} \in \mathbb{R}^D$ are the working states. All streams are initialized identically from h_0 but subsequently follow different propagation and update rules. A DCNO unit corresponds to one operator transformation, such as an attention or feed-forward sublayer, rather than necessarily an entire backbone block. The stream operations act independently at each latent position, while the underlying operator transformation may couple information across positions. $X_l \in \mathbb{R}^{N \times D}$ denotes the collection of read representations over N latent positions.

Dynamic read and operator transformation. At DCNO unit l, let $a_l(H_l) \in \mathbb{R}^r$ denote the state-dependent read coefficients. The read representation at each latent position is constructed as

$$x_l = H_l^\top a_l(H_l) = a_{l,0}(H_l) h_l^{\text{mem}} + \sum_{s=1}^{r-1} a_{l,s}(H_l) h_l^{(s)}.$$

The memory and working states therefore participate in the same learned read mechanism, allowing each unit to adaptively determine how much information to retrieve from each stream. Importantly, the memory read coefficient $a_{l,0}(H_l)$ is independent of the memory write scale introduced below, so the model can make flexible use of the memory state without requiring it to evolve at the same rate.

Collecting the read representations over all N latent positions gives $X_l \in \mathbb{R}^{N \times D}$, which is processed by the operator transformation $Y_l = \mathcal{T}_l(X_l; \mathcal{C}_l) \in \mathbb{R}^{N \times D}$. Here, $\mathcal{T}_l$ denotes the underlying operator sublayer, such as attention, cross-attention, or a feed-forward network, while $\mathcal{C}_l$ denotes its associated context features when applicable. We denote the output at the position under consideration by $y_l \in \mathbb{R}^D$. The operator transformation is applied once to the dynamically aggregated representation rather than independently to each latent stream.

Protected identity path and scaled memory write. Let $P_l(H_l) \in \mathbb{R}^{r \times r}$ denote the learned dynamic-mixing matrix and $b_l(H_l) \in \mathbb{R}^r$ the dynamic-write coefficients at DCNO unit l. We use $P_{l,js}$ to denote the contribution from source stream j to destination stream s, such that unrestricted stream propagation is given by $P_l(H_l)^\top H_l$. For the working states, DCNO retains this learned dynamic mixing, while the memory state is propagated through a protected identity path. Specifically, with $e_0 = (1, 0, \dots, 0)^\top$,

$$\bar{P}_{l,:,0}(H_l) = e_0, \qquad \bar{P}_{l,:,s}(H_l) = P_{l,:,s}(H_l), \quad s = 1, \dots, r-1.$$

To control the evolution rate of the memory state, we further scale its dynamic write. Let $\epsilon_{\text{mem}} \in [0, 1)$ denote the memory write scale and define $D_\epsilon = \text{diag}(\epsilon_{\text{mem}}, 1, \dots, 1)$. The complete state update is $H_{l+1} = \bar{P}_l(H_l)^\top H_l + \big(D_\epsilon b_l(H_l)\big) y_l^\top$. Equivalently, for $s = 1, \dots, r-1$,

$$h_{l+1}^{\text{mem}} = h_l^{\text{mem}} + \epsilon_{\text{mem}} b_{l,0}(H_l) y_l, \qquad h_{l+1}^{(s)} = \sum_{j=0}^{r-1} P_{l,js}(H_l) h_l^{(j)} + b_{l,s}(H_l) y_l. \tag{1}$$

Thus, the memory state preserves its previous representation through the identity path and receives only a scaled write, whereas the working states can undergo dynamic cross-stream mixing and full writes. We use $\epsilon_{\mathrm{mem}} = 0.05$ by default. We note that the protected identity path prevents dynamic mixing from directly overwriting the memory state, while preserving its interaction with the working states. Memory can still contribute to the dynamic read and to working-state mixing through $P_{l,0s}$ for $s \geq 1$, while the working states can influence memory through the subsequent scaled write.

State-dependent connection coefficients. The dynamic read, mixing, and write coefficients are generated from the current latent state. Let $\widetilde{H}_l = \mathrm{Norm}_l(H_l) \in \mathbb{R}^{r\times D}$ denote the normalized state. For compactness, we denote $A_l(H_l) = [a_l(H_l), P_l(H_l)]$. The coefficients are parameterized by

$$A_l(H_l) = A_l^{\mathrm{static}} + \gamma_{\alpha,l}\tanh\big(\widetilde{H}_l W_{\alpha,l}\big), \qquad b_l(H_l) = b_l^{\mathrm{static}} + \gamma_{\beta,l}\tanh\big(\widetilde{H}_l w_{\beta,l}\big).$$

Here, $A_l^{\mathrm{static}} \in \mathbb{R}^{r\times(r+1)}$ and $b_l^{\mathrm{static}} \in \mathbb{R}^r$ are learned static coefficients, while $W_{\alpha,l} \in \mathbb{R}^{D\times(r+1)}$, $w_{\beta,l} \in \mathbb{R}^D$, and $\gamma_{\alpha,l}, \gamma_{\beta,l} \in \mathbb{R}$ parameterize their state-dependent adjustments. We use $\tanh$ rather than sigmoid because these quantities are unconstrained connection coefficients rather than gating probabilities, and therefore need not be nonnegative or sum to one. The coefficients are computed from $\widetilde{H}_l$, while dynamic read and mixing operate on H_l itself; memory-specific protection and write scaling are applied afterward.

Relation to residual connections and hyper-connections (HC). Standard residual connections propagate and update a single evolving state, while HC extends this computation to dynamically interacting streams without a dedicated memory role. On the contrary, DCNO retains the multi-stream formulation but protects one memory destination from incoming mixing and scales its writes. The working streams retain dynamic mixing and unscaled writes, allowing memory and working representations to follow distinct propagation rules.

Appendix B further analyzes the structural properties of the protected memory path, including finite-depth retention bounds and the ability to recover standard residual computation as a special case.

3.2 Context Conditioning and Stream Reduction

We construct DCNO following the attention and feed-forward transformations of GNOT (Hao et al., 2023), a transformer-based neural operator for general PDE learning. Each transformation forms one DCNO unit, and the multi-stream latent state is propagated sequentially through these units with unit-specific connection parameters shared across query locations.

Context Encoding and Conditioning. The query encoder produces the initial representation h_0 at each query location, which initializes the memory and working states as defined above. A separate context encoder maps physical inputs, such as boundary conditions and geometry, to context features. The context encoder and the corresponding conditioning pathway are kept unchanged from the GNOT backbone. These context features are supplied to context-using operator transformations rather than replicated into additional latent streams.

For a cross-attention unit, the dynamically read representation X_l provides the query-side input:

$$Q_l = \mathcal{Q}_l(X_l), \qquad Y_l = \mathrm{CrossAttn}_l(Q_l, K_l, V_l),$$

where $\mathcal{Q}_l$ denotes query projection and preprocessing, while K_l and V_l are constructed from the encoded context. The resulting operator output updates the latent streams according to the rules defined above. For $\epsilon_{\mathrm{mem}} > 0$, separately encoded physical conditions can influence memory through the scaled write, without requiring all such information to be contained in its initialization. Context features also remain directly available to subsequent context-using units as latent states evolve.

Stream Reduction and Prediction. After the final DCNO unit, the latent streams are aggregated at each query location and decoded into the predicted field:

$$x_{\mathrm{out}} = c_r \sum_{s=0}^{r-1} h_L^{(s)}, \qquad \widehat{u} = \mathcal{D}(X_{\mathrm{out}}),$$

Table 1: Comparison of prediction errors across benchmark datasets. Lower is better. The best results are shown in **bold**, and the second-best results are underlined. "–" indicates that the corresponding method is not applicable to the dataset under the evaluated setting. **Our method (DCNO)** achieves the lowest errors in most cases and ranks among the top two methods in every comparison.

Method	NS2D-c			Inductor			NS2D	Darcy2D	Heat2D	NACA	Elasticity
	u	v	p	B_x	B_y	A_z					
FNO	0.0247	0.0503	0.0191	0.3179	0.4993	0.0313	0.1169	0.0105	–	–	–
UNO	0.0268	0.0554	0.0223	0.3199	0.5002	0.0292	0.1365	0.0124	–	–	–
Galerkin	0.0332	0.0876	0.0354	0.2672	0.4024	0.0175	0.1812	0.0305	0.0662	–	–
OFormer	0.0098	0.0223	0.0110	0.2891	0.4206	0.0925	0.1053	0.0285	0.1120	0.0618	**0.0083**
GNOT	0.0102	0.0235	0.0122	0.2572	0.3822	0.0098	0.1002	0.0149	0.0530	0.0253	0.0107
GNOT+HC	0.0087	0.0185	**0.0097**	0.2571	0.3820	0.0101	0.1009	0.0144	0.0432	0.0150	0.0110
LNO	0.0084	0.0205	0.0098	0.3887	0.5704	0.1256	0.1245	0.0081	0.0427	0.0731	0.0159
Transolver	0.0129	0.0290	0.0140	0.2568	0.3806	0.0086	0.1021	0.0141	0.0363	0.0060	0.0304
DCNO	**0.0079**	**0.0183**	0.0098	**0.2563**	**0.3797**	**0.0073**	**0.0893**	**0.0064**	**0.0358**	**0.0058**	0.0093

where X_{out} collects the aggregated representations over all query locations, and $\mathcal{D}$ is the output decoder. The fixed factor c_r is 1 for sum reduction and $1/r$ for mean reduction. Memory and working states participate in the same reduction, with no separate memory decoder or additional prediction head. All streams are initialized anew for each forward pass; memory persistence refers to propagation across DCNO units, rather than physical-time evolution or storage across samples.

4 Experiments

Benchmarks and baselines We evaluate the performance of DCNO on various datasets, including NS2D-c, Inductor, NS2D, Darcy2D, Heat2D, NACA, and Elasticity (Li et al., 2020; 2023b; Hao et al., 2023), which span fluid dynamics, electromagnetics, heat transfer, and elasticity with scalar or multi-component outputs. We compare DCNO against FNO (Li et al., 2020), UNO (Rahman et al., 2022), Galerkin Transformer (Cao, 2021), OFormer (Li et al., 2022), GNOT (Hao et al., 2023), LNO (Wang & Wang, 2024), and Transolver (Wu et al., 2024) to assess its predictive performance relative to representative neural-operator architectures. Among them, GNOT serves as the primary architectural baseline: DCNO retains its attention and feed-forward transformations while replacing residual latent propagation with the proposed multi-stream connections. The remaining methods provide a broader comparison across the evaluated benchmark settings. Detailed descriptions of the datasets are provided in Appendix A, while the baseline methods are introduced in Section 2.

Implementation. The main experiments use a latent width of 128 and four attention heads with linear attention. DCNO uses $r = 4$ streams by default (one memory stream and three working streams) with memory write scale $\epsilon_{\text{mem}} = 0.05$. All models are trained for 500 epochs using AdamW (Loshchilov & Hutter, 2017), an initial learning rate of 10^{-3}, weight decay of 5×10^{-5}, a cosine learning-rate schedule (Loshchilov & Hutter, 2016), batch size 4, and a gradient-clipping threshold of 1000. The separate depth study evaluates NS2D at operator depths $L \in \{1, \ldots, 8\}$ and forecast leads $d \in \{6, 20, 40\}$.

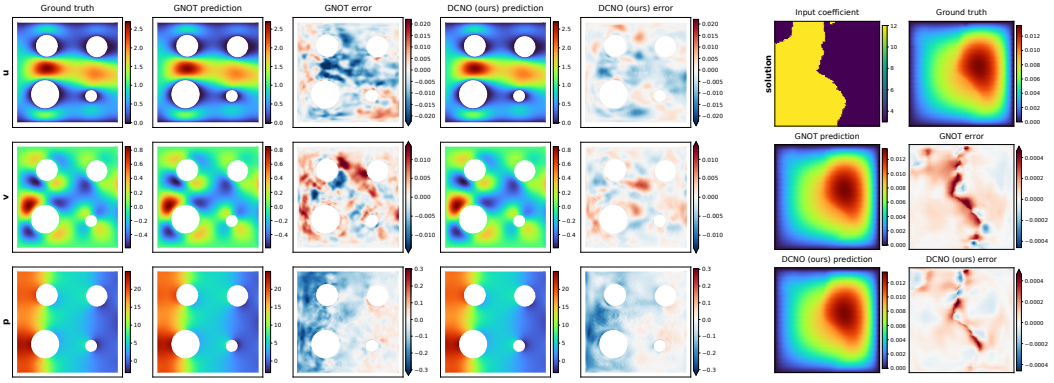
Prediction metrics. For a predicted field $\hat{u}$ and reference field u, we use relative L_2 error,

$$\text{RelL2}(\hat{u}, u) = \|\hat{u} - u\|_2 / \|u\|_2. \tag{2}$$

For NS2D-c and Inductor, Table 1 reports the errors of individual output components. All entries are displayed to four decimal places. Relative error reductions quoted below are approximate, computed from these displayed values as $(E_{\text{baseline}} - E_{\text{DCNO}})/E_{\text{baseline}}$.

4.1 Overall Prediction Results

Comparison with the full baseline set. Compared with the full set of eight baselines reported in Table 1, DCNO achieves the lowest displayed error in nine of the 11 setting comparisons. The



(a) NS2D-c. (b) Darcy2D.

Figure 3: Visualization of the reference fields, predictions, and signed errors on representative NS2D-c and Darcy2D test samples. Corresponding panels use matched field and error scales across methods for each output, with colors closer to gray indicating smaller error magnitudes.

advantage is particularly clear on Darcy2D, NS2D, and Inductor A_z, where DCNO improves over the strongest alternative by approximately 21.0%, 10.9%, and 15.1%, respectively. DCNO also achieves the lowest reported errors on Heat2D and NACA, with smaller margins of approximately 1.4% and 3.3% over the strongest baseline. Among the remaining comparisons, DCNO ranks second on NS2D-c's pressure component and Elasticity, where GNOT+HC (explained in the next paragraph) and OFormer achieve the lowest errors, respectively. Overall, DCNO delivers consistently strong predictive accuracy across the evaluated benchmark suite, achieving the best result in most comparisons and ranking among the top two in all evaluated settings.

Ablation on the connection design. We compare DCNO with GNOT and GNOT+HC to isolate the effect of the connection design. All three variants use the same GNOT backbone and underlying operator transformations, differing only in how hidden states are propagated across layers. GNOT uses standard residual connections, while GNOT+HC replaces them with standard Hyper-Connections (Zhu et al., 2025), without assigning any stream a dedicated memory role. DCNO instead separates the latent streams into working and memory parts. We note that GNOT+HC and DCNO use the same number of streams, dynamic read mechanism, and training settings, differing in the memory-specific propagation rules.

As shown in Table 1, GNOT+HC improves over the residual GNOT baseline in several settings, showing that standard Hyper-Connections can already benefit neural-operator propagation. DCNO achieves lower errors than GNOT in all 11 evaluated settings and further outperforms GNOT+HC in 10 of them. The improvements are particularly pronounced on Darcy2D and NACA, where the errors decrease from 0.0144 to 0.0064 and from 0.0150 to 0.0058, respectively. Several other settings, including NS2D, Heat2D, Elasticity, and Inductor A_z, also show error reductions of roughly 10%–30%. These results indicate that the gains of DCNO extend beyond standard Hyper-Connections, supporting the contribution of its memory-working state connection design.

4.2 Physical-Condition Analysis

Spatial accuracy around physical-condition-induced structures. Figure 3 compares the spatial error patterns of DCNO and the baseline GNOT under matched field and error scales. The signed error is defined as the prediction minus the ground-truth value at each location, with positive and negative values indicating over- and under-prediction, respectively. On NS2D-c, GNOT exhibits pronounced errors along the boundaries of the circular obstacles, particularly in the velocity fields, while these boundary-localized errors are substantially smaller for DCNO. The pressure field shows the same overall tendency, with DCNO producing weaker errors around the obstacle boundaries. On Darcy2D, the largest GNOT errors form a narrow band that closely follows the discontinuous interface in the input coefficient, whereas DCNO markedly reduces the error magnitude along this interface. These examples show that DCNO more accurately resolves local solution structures

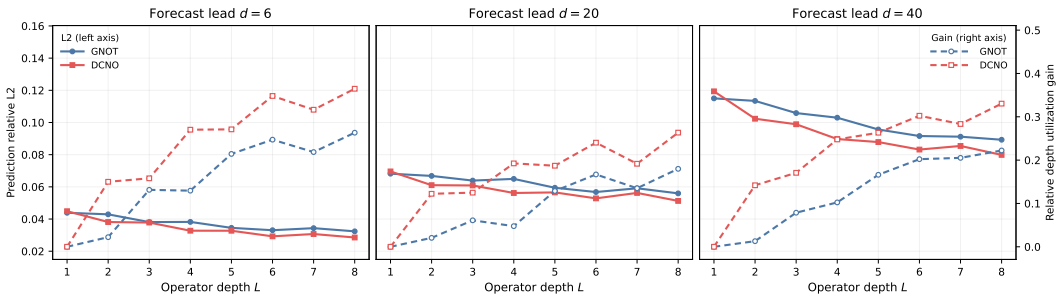


Figure 4: Prediction error and utilization gain across operator depths on NS2D at forecast leads $d = 6, 20, 40$. Solid lines show prediction relative L_2 error (left axis), while dashed lines show the relative depth-utilization gain over each method's one-block model (right axis). DCNO achieves lower error in most multi-block settings and consistently larger gains from increasing depth.

that are strongly shaped by the input physical conditions, highlighting its advantage in capturing condition-dependent solution behavior. We provide additional visualizations in Appendix D.

Alignment with encoded physical conditions across depth. To examine whether physical-condition information remains reflected in the operator representation as depth increases, we measure the alignment between the input to each operator block and the encoded physical conditions using centered linear kernel alignment (CKA) (Kornblith et al., 2019). For each test sample, both representations are summarized by concatenating the spatial mean and standard deviation of each feature channel. Stacking these summaries across samples gives the operator representation matrix X_ℓ and the condition-reference matrix C, and we define

$$P_\ell = \mathrm{CKA}(X_\ell, C) = \frac{\|\widetilde{X}_\ell^\top \widetilde{C}\|_F^2}{\|\widetilde{X}_\ell^\top \widetilde{X}_\ell\|_F \|\widetilde{C}^\top \widetilde{C}\|_F}, \tag{3}$$

where $\widetilde{X}_\ell$ and $\widetilde{C}$ are centered across samples. For GNOT, X_ℓ is the hidden/query state entering the operator block; for DCNO, it is the dynamically read state passed to the operator transformation. Importantly, the same encoded-condition representation C is used as the reference for both models, since DCNO modifies only the query-side latent propagation while leaving the condition encoding unchanged. A larger P_ℓ indicates stronger agreement between the across-sample structure of the operator representation and that of the encoded physical conditions.

Figure 1 reports the mean and ± 1 standard deviation over 20 bootstrap resamples of 100 test samples. GNOT's alignment decreases from approximately 0.4 at the first recorded block to 0.08 at the last, whereas DCNO remains near 0.4 across depth. Since DCNO is evaluated on the read state actually used by the operator transformation, the result shows that its computational representation preserves substantially more condition-related structure across the examined network depth.

4.3 Depth Utilization and Representation Evolution

We next examine whether DCNO makes additional operator blocks more useful for prediction while allowing its operator-input representations to evolve. We evaluate predictive errors and gains across model depths and compare the evolution of memory and read states within trained networks.

Predictive benefit of additional depth. Figure 4 compares the prediction relative errors and gains of DCNO and GNOT on NS2D dataset across $L = 1, \ldots, 8$ operator blocks and forecast leads $d = 6, 20, 40$, where a larger forecast lead corresponds to a longer temporal horizon between the input and target fields. To compare the proportional error reduction from additional blocks relative to each model's one-block baseline, we define the relative depth-utilization gain as

$$G(L, d) = [E(1, d) - E(L, d)] / E(1, d), \tag{4}$$

where $E(L, d)$ is the prediction relative error at depth L and forecast lead d. Thus, $G(1, d) = 0$, and a larger positive value indicates a greater error reduction obtained from additional operator blocks.

DCNO achieves lower prediction errors in most multi-block settings despite slightly higher one-block errors. The advantage also becomes more pronounced at longer forecast leads: at $L = 8$, the error gap between DCNO and GNOT increases from roughly 0.003 at $d = 6$ to about 0.005 at $d = 20$ and 0.009 at $d = 40$. Consistently, DCNO attains larger depth-utilization gains at every tested depth $L > 1$ for all three leads. At $L = 8$, the gains are approximately 36%, 26%, and 33% for $d = 6, 20, 40$, compared with 26%, 18%, and 22% for GNOT. Together, the lower multi-block errors and consistently larger gains show that DCNO makes more effective predictive use of additional operator depth, particularly at the longer forecast horizons examined here.

Memory and state evolution across depth. To further examine the mechanism of DCNO, we track the evolution of the internal latent states of DCNO and the baseline GNOT on the Heat2D using *initial-state alignment* (ISA). ISA measures the cosine similarity between a recorded latent state z_l and its initial reference h_0:

$$\mathrm{ISA}_l(z) = \frac{\langle z_l, h_0 \rangle}{\|z_l\|_2 \, \|h_0\|_2}. \tag{5}$$

Higher ISA indicates stronger directional similarity to the initial representation. We note that ISA is evaluated across connection depth within a fixed trained model, rather than across separately trained models of different depths. Unlike CKA, which compares across-sample structural alignment between latent representations and encoded physical conditions, ISA measures the directional similarity of a latent state to its own initialization.

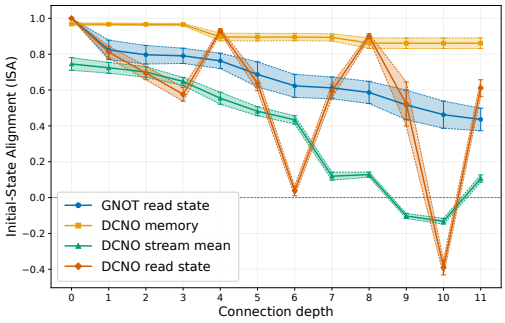


Figure 5: Example of the internal mechanism of DCNO revealed by initial-state alignment (ISA) on Heat2D. DCNO maintains a highly aligned memory state while its stream-mean and read representations evolve substantially, in contrast to GNOT's single evolving state.

Figure 5 provides an example of these state trajectories. For GNOT, the read state (blue curve) progressively departs from the initialization, with ISA decreasing from 1.00 to approximately 0.43. In contrast, DCNO exhibits distinct behaviors across the three measured states. The memory state (h_l^{mem}, yellow curve) remains strongly aligned with its initial reference throughout the network, decreasing only from approximately 0.97 to 0.86. The stream mean (mean representation across all latent streams, green curve) changes much more substantially, approaching zero and becoming negative at later connection sites. Meanwhile, the read state (x_l, dynamically aggregated state, orange curve) varies strongly and nonmonotonically, ranging from values close to 0.9 to approximately -0.4. Taken together, these trajectories show that, whereas GNOT couples information persistence and representation evolution within a single latent stream, DCNO separates a directionally stable memory path from substantially evolving multi-stream and read representations. In particular, the large variation of the read state is consistent with its role as a dynamically recomputed combination of the latent streams rather than a persistent state itself. Thus, DCNO can maintain a stable memory reference without restricting the evolution of the representations used for operator computation.

5 Conclusion

We propose the Dynamic Connection Neural Operator (DCNO) to address the tension between information persistence and representation evolution in deep neural operators. DCNO introduces interacting latent streams with a protected operator-memory path and dynamically evolving working states. Experiments across diverse PDE benchmarks show improved predictive performance, while representation and depth analyses indicate stronger alignment with encoded physical conditions and more effective use of additional operator blocks. These results support separating memory and working streams as an effective design for improving information propagation across operator depth.

Limitations. Our evaluation instantiates DCNO with the GNOT backbone on established PDE benchmarks. Further evaluation on other neural-operator architectures and larger-scale physical systems would help assess its broader applicability.

AI USE STATEMENT

In this work, we used generative AI tools to assist with polishing the writing of the manuscript, including improving grammar, clarity, conciseness, and organization. We have not used generative AI tools for research ideation, methodology development, mathematical derivations, experimental design, data generation, or result interpretation, and the remaining required-disclosure tasks are not applicable to this work. All AI-assisted revisions were reviewed by the authors to ensure technical accuracy and consistency with the intended meaning. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work develops a general neural-operator architecture for learning solution mappings of partial differential equations and does not involve human-subject experiments or the collection of personal or sensitive data. Our empirical evaluation uses established PDE benchmark datasets from prior work and does not introduce or release new datasets. DCNO is intended as a general-purpose method for scientific machine learning, and the ethical considerations of downstream applications depend on the corresponding physical systems, data, and deployment settings. We encourage users to follow applicable requirements regarding data licensing, safety, validation, and responsible deployment, particularly when neural-operator predictions are used in safety-critical scientific or engineering applications.

REPRODUCIBILITY STATEMENT

We provide the methodological, experimental, and theoretical details necessary to reproduce the main results of this work. The DCNO architecture and its multi-stream connection mechanism are specified in Section 3, with a precise forward pass and analyses of parameter, computation, storage, and structural properties provided in Appendix C. The benchmark datasets and governing equations are described in Appendix A, while the experimental settings, evaluation metrics, and main comparisons are reported in Section 4. The theoretical properties of the protected operator-memory mechanism, including finite-depth retention and residual-emulation results, are developed in Appendix B. We plan to release the implementation and experiment configurations upon publication.

REFERENCES

Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6(5):320–328, 2024.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in artificial intelligence*, pp. 1352–1361. PMLR, 2021.

Susanne C Brenner and L Ridgway Scott. *The mathematical theory of finite element methods*. Springer, 2008.

Shuhao Cao. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, 34:24924–24940, 2021.

Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. *Advances in neural information processing systems*, 30, 2017.

Lawrence C Evans. *Partial differential equations*, volume 19. American mathematical society, 2022.

Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. *Advances in neural information processing systems*, 34:24048–24062, 2021.

Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. Gnot: A general neural operator transformer for operator learning. In *International conference on machine learning*, pp. 12556–12569. PMLR, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 2261–2269. Ieee, 2017.

Pengzhan Jin, Shuai Meng, and Lu Lu. Mionet: Learning multiple-input operators via tensor product. *SIAM Journal on Scientific Computing*, 44(6):A3490–A3514, 2022.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMlR, 2019.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022.

Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable transformer for pde surrogate modeling. *Advances in Neural Information Processing Systems*, 36:28010–28039, 2023a.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023b.

Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, 36:35836–35854, 2023c.

Songming Liu, Zhongkai Hao, Chengyang Ying, Hang Su, Ze Cheng, and Jun Zhu. Nuno: A general framework for learning parametric pdes with non-uniform data. In *International Conference on Machine Learning*, pp. 21658–21671. PMLR, 2023.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for pdes on million-scale geometries. *arXiv preprint arXiv:2502.02414*, 2025.

Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. *Advances in neural information processing systems*, 37:136479–136508, 2024.

Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-no: U-shaped neural operators. *arXiv preprint arXiv:2204.11127*, 2022.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. *Advances in neural information processing systems*, 28, 2015.

Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, et al. Attention residuals. *arXiv preprint arXiv:2603.15031*, 2026.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 32–42. IEEE, 2021.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. *arXiv preprint arXiv:2111.13802*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6761–6774, 2024.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 1810–1822, 2019.

Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse pde problems. *Advances in Neural Information Processing Systems*, 37:33085–33107, 2024.

Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

Haixu Wu, Tengge Hu, Huakun Luo, Jianmin Wang, and Mingsheng Long. Solving high-dimensional pdes with latent spectral models. *arXiv preprint arXiv:2301.12664*, 2023.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

Yanming Xia and Angelica I Aviles-Rivero. Do neural operators forget geometry? the forgetting hypothesis in deep operator learning. *arXiv preprint arXiv:2605.05862*, 2026.

Zipeng Xiao, Zhongkai Hao, Bokai Lin, Zhijie Deng, and Hang Su. Improved operator learning by orthogonal attention. *arXiv preprint arXiv:2310.12487*, 2023.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, et al. mhc: Manifold-constrained hyper-connections. *arXiv preprint arXiv:2512.24880*, 2025.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *International Conference on Learning Representations*, volume 2025, pp. 97183–97219, 2025.

A Dataset Details and Governing Equations

We describe the seven benchmark families in Section 4, using the dataset names of the main text. Elasticity and NACA2D originate from Geo-FNO (Li et al., 2023b); Darcy2D and the time-dependent Navier–Stokes data originate from FNO (Li et al., 2020); Heat2D, Inductor2D, and the flow around four circular obstacles were introduced with GNOT (Hao et al., 2023). The governing equations, physical inputs, target fields, and reference dataset configurations are summarized below. Throughout this section, $\boldsymbol{x} = (x, y)$ denotes a spatial coordinate, $\boldsymbol{n}$ an outward unit normal, and $\mathbf{I}$ an identity tensor.

Table 2: Benchmark inputs and outputs.

Dataset	Physical inputs	Target field(s)
Elasticity	Geometry of a solid with a central void	Scalar stress field
Heat2D	Two internal spline boundaries; top-boundary temperature	Temperature T
Inductor2D	Geometry, coil current, and relative permeability	B_x, B_y, A_z
NACA2D	Airfoil geometry represented by mesh coordinates	Mach number M
NS2D-4balls	Geometry of four circular obstacles	Velocity u_x, u_y; pressure p
NS2D	Earlier vorticity observations	Later vorticity ω
Darcy2D	Spatial permeability coefficient a	Scalar solution u

A.1 ELASTICITY

The Geo-FNO elasticity benchmark concerns an incompressible hyperelastic material occupying a unit square with a central, irregular void. The lower edge is clamped and the upper edge carries traction $(0, 100)^\mathsf{T}$ in the source's units. The void radius varies between 0.2 and 0.4 (Li et al., 2023b, Section 4.1). For a static displacement field $\boldsymbol{d}$, force balance is

$$\nabla\cdot\boldsymbol{\sigma} = \mathbf{0} \quad \text{in } \Omega, \qquad \boldsymbol{d} = \mathbf{0} \quad \text{on } \Gamma_{\text{bottom}}, \qquad \boldsymbol{\sigma}\boldsymbol{n} = (0, 100)^\mathsf{T} \quad \text{on } \Gamma_{\text{top}}, \tag{6}$$

where $\boldsymbol{\sigma}$ denotes the stress tensor. This is the equilibrium form of the solid momentum equation given in the source. The reported Rivlin–Saunders strain-energy density is

$$W = C_1(I_1 - 3) + C_2(I_2 - 3), \qquad I_1 = \operatorname{tr}\mathbf{C}, \qquad I_2 = \tfrac{1}{2}\big[(\operatorname{tr}\mathbf{C})^2 - \operatorname{tr}(\mathbf{C}^2)\big], \tag{7}$$

where $\mathbf{C}$ is the right Cauchy–Green tensor and C_1, C_2 are material constants (Li et al., 2023b). The original simulations use finite elements and a $1000/200$ training/test split. Geo-FNO's point-cloud comparison reports 972 input points per sample. The learning task maps geometry to the stress field (Li et al., 2023b, Section 4.1 and Table 1).

A.2 HEAT2D

GNOT's Heat dataset uses $\Omega = [0, 9]^2$, partitioned into three material regions by two splines. The inputs specify these interfaces and the temperature on the upper boundary. Its steady equation is (Hao et al., 2023, Appendix A)

$$\rho C_p \boldsymbol{u}\cdot\nabla T - k\Delta T = Q, \qquad T(x, 9) = g_{\text{top}}(x), \qquad T(0, y) = T(9, y), \tag{8}$$

with lateral periodicity. Here ρ, C_p, k, $\boldsymbol{u}$, and Q are density, specific heat capacity, conductivity, transport velocity, and heat-source density, respectively; T is the predicted temperature. The source reports COMSOL Multiphysics 6.0 simulations and releases of 1100 and 5500 samples (Hao et al., 2023).

A.3 INDUCTOR2D

This GNOT benchmark varies inductor geometry, coil current I_{coil}, and relative permeability μ_r, predicting B_x, B_y, A_z. Its steady electromagnetic system is (Hao et al., 2023, Appendix A, Figures 6–7)

$$\begin{aligned} \nabla\times\boldsymbol{H} &= \boldsymbol{J}, & \boldsymbol{B} &= \nabla\times\boldsymbol{A}, \\ \boldsymbol{B} &= \mu_0\mu_r\boldsymbol{H}, & \boldsymbol{J} &= \sigma_{\text{el}}\boldsymbol{E} + \sigma_{\text{el}}\boldsymbol{v}\times\boldsymbol{B} + \boldsymbol{J}_e, \end{aligned} \tag{9}$$

$$\boldsymbol{n}\times\boldsymbol{A} = \mathbf{0} \quad \text{on } \partial\Omega, \qquad \boldsymbol{J}_e = \frac{NI_{\text{coil}}}{S_{\text{coil}}}\boldsymbol{e}_{\text{coil}}. \tag{10}$$

Here $\boldsymbol{H}, \boldsymbol{B}, \boldsymbol{A}, \boldsymbol{J}, \boldsymbol{E}$ denote magnetic field strength, magnetic flux density, magnetic vector potential, current density, and electric field. Also, σ_{el} is conductivity, μ_0 vacuum permeability, $\boldsymbol{v}$ material velocity, N the coil turn count, S_{coil} coil cross-sectional area, and $\boldsymbol{e}_{\text{coil}}$ the current direction. The COMSOL-generated reference set contains 1000 training and 100 test samples (Hao et al., 2023).

A.4 NACA2D

The Geo-FNO airfoil data describe steady transonic, inviscid flow. The time-dependent conservation equations used to obtain the steady solution are (Li et al., 2023b, Section 4.4)

$$\begin{aligned} \partial_t \rho + \nabla\cdot(\rho \boldsymbol{u}) &= 0, \\ \partial_t(\rho \boldsymbol{u}) + \nabla\cdot(\rho \boldsymbol{u}\otimes\boldsymbol{u} + p\mathbf{I}) &= \mathbf{0}, \\ \partial_t E + \nabla\cdot\big[(E+p)\boldsymbol{u}\big] &= 0, \end{aligned} \tag{11}$$

where ρ, $\boldsymbol{u}$, p, and E are density, velocity, pressure, and total energy per unit volume. The ideal-gas closure is

$$p = (\gamma-1)\left(E - \tfrac{1}{2}\rho\|\boldsymbol{u}\|^2\right), \qquad c = \sqrt{\gamma p/\rho}, \qquad \gamma = 1.4, \tag{12}$$

where γ is the ratio of specific heats and c the local sound speed. The time derivatives vanish at steady state. The source imposes $\rho_\infty = p_\infty = 1$, far-field Mach number $M_\infty = 0.8$, zero angle of attack, and no penetration $\boldsymbol{u}\cdot\boldsymbol{n} = 0$ at the airfoil surface. Starting from NACA-0012, a cubic design element with eight control nodes parameterizes shape changes, with vertical displacements drawn from $\mathcal{U}[-0.05, 0.05]$. A second-order implicit finite-volume solver supplies the reference solutions on a C-grid with approximately 200×50 quadrilateral elements. The reference dataset uses 1000 training and 200 test samples, with mesh coordinates as inputs and the Mach-number field $M = \|\boldsymbol{u}\|/c$ as the target (Li et al., 2023b, Section 4.4).

A.5 NS2D-4BALLS

This task corresponds to GNOT's NS2d-c problem on $\Omega = [0,8]^2 \setminus \bigcup_{i=1}^4 R_i$, where R_i are circular obstacles with varying positions. The steady incompressible equations and boundary data are (Hao et al., 2023, Appendix A)

$$\begin{aligned} (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = -\nabla p + \mathrm{Re}^{-1}\Delta\boldsymbol{u}, \quad \nabla\cdot\boldsymbol{u} = 0 \quad & \text{in } \Omega, \\ \boldsymbol{u} = \mathbf{0} \quad & \text{on solid walls}, \\ u_x = y(8-y)/16 \quad & \text{at the inlet}, \\ p = 0 \quad & \text{at the outlet}. \end{aligned} \tag{13}$$

Here Re is the Reynolds number. The targets are (u_x, u_y, p). GNOT reports $1000/100$ training/test samples generated with COMSOL (Hao et al., 2023).

A.6 NS2D

The time-dependent FNO benchmark evolves scalar vorticity ω on the unit torus, with periodic boundary conditions (Li et al., 2020):

$$\begin{aligned} \partial_t\omega + \boldsymbol{u}\cdot\nabla\omega = \nu\Delta\omega + f, \qquad & \boldsymbol{x}\in(0,1)^2, \quad t>0, \\ \nabla\cdot\boldsymbol{u} = 0, \qquad & \omega = \partial_x u_y - \partial_y u_x, \\ \omega(\boldsymbol{x},0) = \omega_0(\boldsymbol{x}). & \end{aligned} \tag{14}$$

Here ν is kinematic viscosity. FNO specifies the forcing and initial-field law as

$$\begin{aligned} f(x,y) &= 0.1\big[\sin(2\pi(x+y)) + \cos(2\pi(x+y))\big], \\ \omega_0 &\sim \mathcal{N}\Big(0, 7^{3/2}(-\Delta + 49\mathcal{I})^{-5/2}\Big), \end{aligned} \tag{15}$$

where $\mathcal{I}$ is the identity operator and the covariance uses the periodic Laplacian. The published generator uses a dealiased pseudospectral method, Crank–Nicolson time integration with solver step 10^{-4}, and 256×256 grids, subsequently downsampled to 64×64. Stored snapshots are separated by one time unit (Li et al., 2020, Appendix A.3.3). The source includes several viscosity/horizon configurations; the benchmark name alone does not select one. The present depth study reports forecast leads $d \in \{6, 20, 40\}$. These leads should be distinguished from the numerical solver's integration step.

A.7 DARCY2D

Darcy2D learns the permeability-to-solution map $a \mapsto u$ for (Li et al., 2020)

$$-\nabla \cdot \big(a(\boldsymbol{x})\nabla u(\boldsymbol{x})\big) = 1 \quad \text{in } \Omega = (0,1)^2, \qquad u = 0 \quad \text{on } \partial\Omega. \tag{16}$$

The source constructs a binary coefficient field by thresholding a Gaussian field:

$$g \sim \mathcal{N}\big(0, (-\Delta + 9\mathcal{I})^{-2}\big), \qquad a(\boldsymbol{x}) = \begin{cases} 12, & g(\boldsymbol{x}) > 0, \\ 3, & g(\boldsymbol{x}) < 0. \end{cases} \tag{17}$$

The Laplacian in this sampling law has homogeneous Neumann boundary conditions; the solution u instead has the homogeneous Dirichlet condition in Equation (16). FNO uses a second-order finite-difference solver on 421×421 grids and obtains coarser resolutions by subsampling, with $1000/200$ training/test examples (Li et al., 2020, Section 5 and Appendix A.3.2).

B THEORETICAL PROPERTIES OF PROTECTED OPERATOR MEMORY

We analyze the connection rule in Section 3. Identity-path analysis for residual networks motivates direct propagation across depth (He et al., 2016b), and Hyper-Connections already include residual computation as a special case (Zhu et al., 2025). The results below address the additional requirements of DCNO: protecting one memory destination, scaling its writes, retaining unrestricted working computation, and reducing all streams with the prescribed output rule. The results characterize the protected memory path through conditional retention and sensitivity bounds, together with exact residual-emulation properties.

B.1 NOTATION AND SCOPE

We use $k = 0, \ldots, K-1$ for connection sites, irrespective of their grouping into backbone blocks. Let H_k collect the stream states at all N positions. Vectorize the memory field as $m_k \in \mathbb{R}^{ND}$ and concatenate the working fields as $w_k \in \mathbb{R}^{N(r-1)D}$. Euclidean norms of these vectors equal Frobenius norms of the corresponding fields. With $\epsilon = \epsilon_{\mathrm{mem}}$, the full update can be written

$$m_{k+1} = m_k + \epsilon f_k(m_k, w_k), \qquad w_{k+1} = g_k(m_k, w_k), \tag{18}$$

where f_k collects $b_{k,0}(H_{k,i})y_{k,i}$ over positions i. The functions f_k and g_k include coefficient generation, dynamic reads, and the spatially coupled operator branch. Auxiliary inputs are suppressed; when comparing perturbed trajectories, they are held fixed. No analysis below treats the learned connection coefficients as probabilities.

B.2 FINITE-DEPTH RETENTION AND RECOVERABLE INFORMATION

Proposition B.1 (Memory drift controlled by accumulated writes)**.** *For a fixed DCNO network and $0 \le k < n \le K$,*

$$m_n - m_k = \epsilon \sum_{j=k}^{n-1} f_j(m_j, w_j). \tag{19}$$

Define $B_j = |b_{j,0}^{\mathrm{static}}| + |\gamma_{\beta,j}|$ and let $Y_j \in \mathbb{R}^{N\times D}$ be the branch output. Then

$$\|m_n - m_k\|_2 \le \epsilon \sum_{j=k}^{n-1} B_j \|Y_j\|_F =: \Delta_{k,n}. \tag{20}$$

In particular, if $B_j \le B$ and $\|Y_j\|_F \le M$ on this interval, then $\Delta_{k,n} \le \epsilon(n-k)BM$.

Proof. Sum the memory recurrence in (18). Since $|\tanh(t)| \le 1$, the parameterization of $b_{j,0}$ gives $|b_{j,0}(H_{j,i})| \le B_j$ at every position. Therefore $\|f_j\|_2^2 = \sum_i |b_{j,0}(H_{j,i})|^2 \|y_{j,i}\|_2^2 \le B_j^2 \|Y_j\|_F^2$. The triangle inequality proves the bound. □

The identity coefficient prevents skip-path mixing from contributing an additional memory drift term. In an unrestricted multi-stream update, that term would be $\sum_j P_{k,j0} h_k^{(j)} - h_k^{\mathrm{mem}}$ at each position; it need not be small even when the branch output is zero. Ordinary residual connections also have an identity skip. DCNO's distinction is to combine a protected, scaled-write destination with working streams that can continue to mix and transform.

Theorem B.2 (Conditional retention of encoded information)**.** *Let $\delta_n = \Delta_{0,n}$. The following statements hold for the trajectory of a fixed network.*

1. *If $m_0 \neq 0$ and $\delta_n < \|m_0\|_2$, then $m_n \neq 0$ and*

$$\frac{\langle m_n, m_0 \rangle}{\|m_n\|_2 \|m_0\|_2} \geq \sqrt{1 - \left(\frac{\delta_n}{\|m_0\|_2}\right)^2}. \tag{21}$$

2. *Let p be a physical descriptor already recoverable from the initial memory by a map ψ, with $\|\psi(m_0) - p\| \leq \eta$. If ψ is L_ψ-Lipschitz on a set containing m_0 and m_n, then*

$$\|\psi(m_n) - p\| \leq \eta + L_\psi \delta_n. \tag{22}$$

3. *For two inputs with initial memories $m_0, \widetilde{m}_0$ and respective drift bounds $\delta_n, \widetilde{\delta}_n$,*

$$\|m_n - \widetilde{m}_n\|_2 \geq \|m_0 - \widetilde{m}_0\|_2 - \delta_n - \widetilde{\delta}_n. \tag{23}$$

Thus an initial separation greater than the sum of the drift bounds cannot collapse to zero at that depth.

Proof. Proposition B.1 places m_n in the ball of radius δ_n around m_0. Write $a = \|m_0\|_2$ and let ϑ be the angle between the two vectors. Since $\langle m_n, m_0 \rangle \geq a(a - \delta_n) > 0$, the angle is acute. The distance from m_0 to the line spanned by m_n is $a \sin \vartheta$, which is no greater than $\|m_n - m_0\|_2 \leq \delta_n$. This proves (21). For the second claim, $\|\psi(m_n) - p\| \leq \|\psi(m_n) - \psi(m_0)\| + \|\psi(m_0) - p\|$; apply Lipschitz continuity and the drift bound. For the third claim, apply the triangle inequality to $m_0 - \widetilde{m}_0 = (m_0 - m_n) + (m_n - \widetilde{m}_n) + (\widetilde{m}_n - \widetilde{m}_0)$ and rearrange. □

This gives a precise connection between small latent drift and preservation of information that was initially encoded. The auxiliary recovery map ψ is a theoretical certificate, not an added network decoder. The retention result applies to descriptors recoverable from the initial memory. For a desired drift tolerance τ, the sufficient condition $\epsilon \sum_{j<n} B_j \|Y_j\|_F \leq \tau$ exposes the relevant depth budget. Uniform bounds give $\epsilon n B M \leq \tau$. The retention budget is therefore determined jointly by the write scale, network depth, connection coefficients, and branch-output magnitudes.

B.3 Exact persistence without loss of residual computation

Proposition B.3 (An exact recoverable reference)**.** *Suppose the memory write is identically zero at every site, either because $\epsilon = 0$ or because $b_{k,0} \equiv 0$. For fixed auxiliary inputs, let $\Phi_n : m_0 \mapsto (m_n, w_n)$ be the complete state map with all streams initialized from m_0. Then projection onto the memory stream is a left inverse of Φ_n, and*

$$\|\Phi_n(m_0) - \Phi_n(\widetilde{m}_0)\|_2 \geq \|m_0 - \widetilde{m}_0\|_2. \tag{24}$$

At differentiability points, $(D\Phi_n)^\top D\Phi_n \succeq I$.

Proof. The recurrence gives $m_n = m_0$. The squared full-state distance is the squared memory distance plus a nonnegative squared working-state distance. Also $D\Phi_n = [I; Dw_n]$, so $(D\Phi_n)^\top D\Phi_n = I + (Dw_n)^\top Dw_n \succeq I$. □

The next theorem shows that this exact retention regime need not disable the original operator computation. It also handles the fixed reduction of *all* streams, so no change to the decoder is hidden in the construction.

Theorem B.4 (Residual emulation with protected memory). *Consider any $K \geq 1$ residual sites*

$$Z_{k+1} = Z_k + \mathcal{T}_k(Z_k; \mathcal{C}_k), \qquad Z_0 = M_0, \tag{25}$$

followed by a decoder $\mathcal{D}$. Assume the same branch computations and auxiliary inputs are available to DCNO. For every $r \geq 2$, every $\epsilon \in [0, 1)$, and either $c_r = 1$ or $c_r = 1/r$, there exist legal constant connection coefficients such that

1. *the memory equals M_0 at every site;*
2. *the branch input at site k equals Z_k;*
3. *the reduced final state equals Z_K, and hence the original decoder gives exactly $\mathcal{D}(Z_K)$.*

The construction uses the same K branch evaluations as the residual model.

Proof. Use working stream 1 as the active stream. Set all memory writes and all writes to streams $s \geq 2$ to zero, and propagate those extra streams by identity. They therefore remain equal to M_0. At every site set $a_k = e_1$, so the branch reads only the active stream.

For $k < K - 1$, choose $P_k = I_r$ and $b_{k,1} = 1$. Starting from the required replicated initialization, induction gives $h_k^{(1)} = z_k$ at each position up to the final branch input. At the final site, retain the identity destination columns for all other streams and choose

$$P_{K-1,:,1} = c_r^{-1} e_1 - (r-1)e_0, \qquad b_{K-1,1} = c_r^{-1}. \tag{26}$$

The active output is then $h_K^{(1)} = c_r^{-1} z_K - (r-1)h_0$, while the other $r-1$ streams equal h_0. Consequently

$$c_r \sum_{s=0}^{r-1} h_K^{(s)} = c_r\big(c_r^{-1} z_K - (r-1)h_0 + (r-1)h_0\big) = z_K. \tag{27}$$

This also covers $K = 1$: the final rule is applied directly to the replicated initial state. All prescribed coefficients are realizable by setting $\gamma_{\alpha,k} = \gamma_{\beta,k} = 0$ and assigning the learned static coefficients accordingly. Negative mixing coefficients are allowed by Section 3.1. Memory protection is respected throughout. □

Thus, DCNO contains the residual backbone as a special case while preserving an internal memory reference. This establishes the compatibility of memory retention with the original residual computation.

B.4 Perturbation transfer with dynamic feedback

The scaled write also limits one-step transfer from working states into memory, provided the unscaled write has controlled sensitivity. Unlike a frozen-coefficient argument, the following analysis recomputes all reads, coefficients, and branch outputs along both trajectories. Here, the perturbation is introduced at an intermediate connection site, which is reindexed as $k = 0$.

Proposition B.5 (Coupled finite-depth sensitivity). *For two trajectories with the same network parameters and auxiliary inputs, assume, on a domain containing the compared states, that*

$$\begin{aligned} \|f_k(m, w) - f_k(\widetilde{m}, \widetilde{w})\|_2 &\leq \alpha_k \|m - \widetilde{m}\|_2 + \beta_k \|w - \widetilde{w}\|_2, \\ \|g_k(m, w) - g_k(\widetilde{m}, \widetilde{w})\|_2 &\leq \chi_k \|m - \widetilde{m}\|_2 + \rho_k \|w - \widetilde{w}\|_2, \end{aligned} \tag{28}$$

with nonnegative constants. Define $u_k = \|m_k - \widetilde{m}_k\|_2$, $v_k = \|w_k - \widetilde{w}_k\|_2$, and

$$Q_k = \begin{pmatrix} 1 + \epsilon\alpha_k & \epsilon\beta_k \\ \chi_k & \rho_k \end{pmatrix}. \tag{29}$$

Then, componentwise,

$$\begin{pmatrix} u_n \\ v_n \end{pmatrix} \leq Q_{n-1} \cdots Q_k \begin{pmatrix} u_k \\ v_k \end{pmatrix}. \tag{30}$$

In particular, for a shared current memory ($u_k = 0$), $u_{k+1} \leq \epsilon\beta_k v_k$.

Proof. Subtract the two memory updates and use the triangle inequality to get $u_{k+1} \le (1+\epsilon\alpha_k)u_k + \epsilon\beta_k v_k$. The second assumption gives $v_{k+1} \le \chi_k u_k + \rho_k v_k$. Iterate these inequalities; each Q_k has nonnegative entries and thus preserves componentwise inequalities. □

Corollary B.6 (Weak transfer in a contractive working-state regime)**.** *Suppose the preceding assumptions hold uniformly with constants $\alpha, \beta, \chi, \rho$ and $0 \le \rho < 1$. Put $\theta = \beta/(1-\rho)$ and $C = \alpha + \theta\chi$. Then*

$$u_n \le (1+\epsilon C)^n (u_0 + \epsilon\theta v_0) \le e^{\epsilon C n}(u_0 + \epsilon\theta v_0). \tag{31}$$

For initially equal memories and $v_0 > 0$, the transfer gain u_n/v_0 is $O(\epsilon)$, as long as ϵn, C, and θ remain bounded.

Proof. Set $V_k = u_k + \epsilon\theta v_k$. Since $\beta + \theta\rho = \theta$,

$$\begin{aligned} V_{k+1} &\le [1 + \epsilon(\alpha + \theta\chi)]u_k + \epsilon(\beta + \theta\rho)v_k \\ &= (1+\epsilon C)u_k + \epsilon\theta v_k \le (1+\epsilon C)V_k. \end{aligned}$$

Iterate and use $u_n \le V_n$ and $1 + t \le e^t$ for $t \ge 0$. □

We also note that Corollary B.6 applies to contractive working-state regimes, whereas Proposition B.5 gives the general finite-depth bound.

Accounting for dynamic coefficients in the assumptions. Let $S = (m, w)$, let $\mathcal{R}_k(S)$ be the collected dynamic read, and let $\boldsymbol{b}_{k,0}(S)$ collect the memory-write coefficients over positions. On a bounded domain suppose $\|\boldsymbol{b}_{k,0}\|_\infty \le B_k$, $\|Y_k\|_F \le M_k$, and the read, branch, and coefficient maps have Lipschitz constants $L_{R,k}$, $L_{T,k}$, and $L_{b,k}$, respectively (using the infinity norm for coefficient outputs). The product inequality gives

$$\|f_k(S) - f_k(\widetilde{S})\|_2 \le (B_k L_{T,k} L_{R,k} + M_k L_{b,k})\|S - \widetilde{S}\|_2. \tag{32}$$

Indeed, split the difference into $b(S)[Y(S) - Y(\widetilde{S})]$ and $[b(S) - b(\widetilde{S})]Y(\widetilde{S})$ and bound each term. If the connection normalization is $L_{N,k}$-Lipschitz in Frobenius norm, then $L_{b,k} \le |\gamma_{\beta,k}|\|w_{\beta,k}\|_2 L_{N,k}$ follows from the 1-Lipschitz property of $\tanh$. This bound accounts for the state dependence of both the connection coefficients and the operator outputs.

C Algorithmic Details and Structural Properties

C.1 A precise forward pass

For clarity, $A_k = [a_k, P_k] \in \mathbb{R}^{r\times(r+1)}$: its first column specifies the read, and its remaining r columns specify propagation. In $P_{k,js}$, j is the source stream and s is the destination. Thus protection replaces the memory *destination column* with e_0, while retaining the memory source row for other destinations. The algorithm below suppresses the batch dimension and uses $H_{k,i} \in \mathbb{R}^{r\times D}$ at position i.

DCNO forward pass at the existing residual sites

1. Encode the original query path to obtain $Z_0 \in \mathbb{R}^{N\times D}$. Initialize $H_{0,i} = \mathbf{1}_r z_{0,i}^\top$ for every position i. Construct external conditioning branches as in the backbone.
2. For $k = 0, \dots, K-1$, perform the following operations using the same, unchanged input state H_k:

$$\begin{aligned} \widetilde{H}_{k,i} &= \mathrm{Norm}_k(H_{k,i}), \\ [a_{k,i}, P_{k,i}] &= A_k^{\mathrm{static}} + \gamma_{\alpha,k}\tanh(\widetilde{H}_{k,i}W_{\alpha,k}), \\ b_{k,i} &= b_k^{\mathrm{static}} + \gamma_{\beta,k}\tanh(\widetilde{H}_{k,i}w_{\beta,k}), \\ x_{k,i} &= H_{k,i}^\top a_{k,i}. \end{aligned}$$

Collect the reads into X_k and evaluate the branch once: $Y_k = \mathcal{T}_k(X_k; \mathcal{C}_k)$. Set $\bar{P}_{k,i,:,0} = e_0$ and $\bar{P}_{k,i,:,s} = P_{k,i,:,s}$ for $s \geq 1$, then update all streams simultaneously:

$$H_{k+1,i} = \bar{P}_{k,i}^{\top} H_{k,i} + (D_\epsilon b_{k,i}) y_{k,i}^{\top}.$$

3. Reduce the final streams as $x_{\mathrm{out},i} = c_r H_{K,i}^{\top} \mathbf{1}_r$ and apply the original decoder to the collected field X_{out}.

In-place updates of individual streams before the remaining reads or coefficients have been evaluated would implement a different recurrence. Normalization generates connection coefficients; the read and propagation use the unnormalized $H_{k,i}$. Branch normalization, if present, remains inside $\mathcal{T}_k$. At a cross-attention site, the read changes the query-side input, while external keys and values retain their original construction. No separate attention or feed-forward branch is evaluated for the memory stream. The memory state is reinitialized for every input and is not a cache across samples or an autoregressive time integrator.

C.2 Parameter, arithmetic, and storage costs

The coefficient shapes in Section 3.1 give the following count per connection site, excluding normalization parameters and the backbone:

$$\begin{aligned} p_{\mathrm{conn}} &= \underbrace{r(r+1)}_{A^{\mathrm{static}}} + \underbrace{D(r+1)}_{W_\alpha} + \underbrace{r}_{b^{\mathrm{static}}} + \underbrace{D}_{w_\beta} + \underbrace{2}_{\gamma_\alpha, \gamma_\beta} \\ &= (r+2)(D+r) + 2. \end{aligned} \tag{33}$$

For example, $r = 4$ and $D = 128$ give 794 coefficient parameters per site. With independent parameters at K sites, the count is $K p_{\mathrm{conn}}$ plus the parameters of the connection normalizations. The fixed write scale and the protected identity column introduce no trainable parameters. The r static entries and D projection entries used only to generate the overwritten memory-destination column do not affect the forward map.

Assuming connection normalization costs $O(NrD)$, dense coefficient generation costs $O(NrD(r+2))$, propagation costs $O(Nr^2D)$, and reads and writes cost $O(NrD)$. If the original branch costs $T_k(N, D; \mathcal{C}_k)$, the total arithmetic per site is

$$T_k(N, D; \mathcal{C}_k) + O(Nr^2D). \tag{34}$$

Only one original branch evaluation is required at each site. For fixed r, the additional arithmetic is linear in the number of positions.

Persistent stream storage is $O(NrD)$ rather than $O(ND)$; materializing the dynamic coefficients requires an additional $O(Nr^2)$ values per site. Without checkpointing or recomputation, retaining connection intermediates for all K sites uses $O(K(NrD + Nr^2))$ storage in addition to the original branch activations. Batch size multiplies these field-dependent costs. The site count K should not be identified with the number of backbone blocks unless their residual-site layout has been specified.

C.3 Inheritance of spatial permutation equivariance

Proposition C.1 (Equivariance of the connection wrapper)**.** *Let Π permute the N query positions and let $\Pi_{\mathcal{C}}$ denote the corresponding action on any position-indexed auxiliary inputs. Assume connection parameters are shared over positions, connection normalization commutes with this permutation, and*

$$\mathcal{T}_k(\Pi X; \Pi_{\mathcal{C}} \mathcal{C}) = \Pi \mathcal{T}_k(X; \mathcal{C}). \tag{35}$$

Then the DCNO site map $\mathcal{U}_k$ satisfies

$$\mathcal{U}_k(\Pi \mathsf{H}; \Pi_{\mathcal{C}} \mathcal{C}) = \Pi \mathcal{U}_k(\mathsf{H}; \mathcal{C}). \tag{36}$$

If the encoder and decoder are also equivariant, so is their composition with all DCNO sites and the final stream reduction.

Proof. Shared coefficient projections and elementwise $\tanh$ commute with a permutation of positions, as does the assumed normalization. Thus permuting the state permutes its reads and connection coefficients. By (35), it also permutes the branch output. Replacing the same destination column at each position by e_0, mixing streams, and writing the outer product are all local in the position index, so they commute with Π. This proves the site claim. Replication, stream reduction, and composition of equivariant maps preserve equivariance, proving the final statement. □

The connection design preserves the spatial permutation equivariance of the backbone. Coordinates and other position-associated features must be reindexed consistently; spatial permutations do not permute the distinct memory and working roles.

D Additional Field Visualizations

We provide additional qualitative comparisons between GNOT and DCNO on Elasticity, Heat2D, and NS2D. Figure 6 shows predicted stress fields and signed errors for two representative Elasticity cases, while Figure 7 presents temperature predictions and errors on Heat2D. Figure 8 further visualizes vorticity predictions at multiple forecast leads on NS2D. Together, these examples complement the quantitative results in Table 1 with spatial comparisons of the predicted fields and corresponding errors.

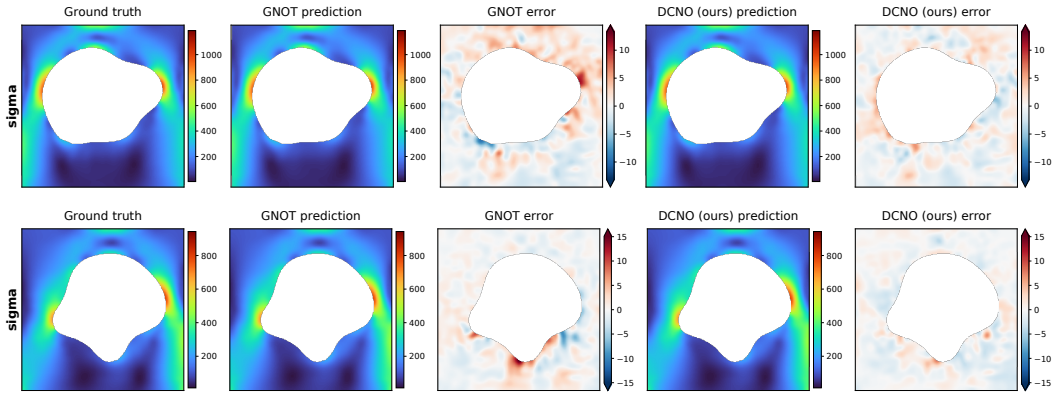


Figure 6: Additional qualitative results on Elasticity. Each row shows the reference stress field, GNOT prediction and signed error, and DCNO prediction and signed error. Field scales and symmetric error scales are matched between methods within each row.

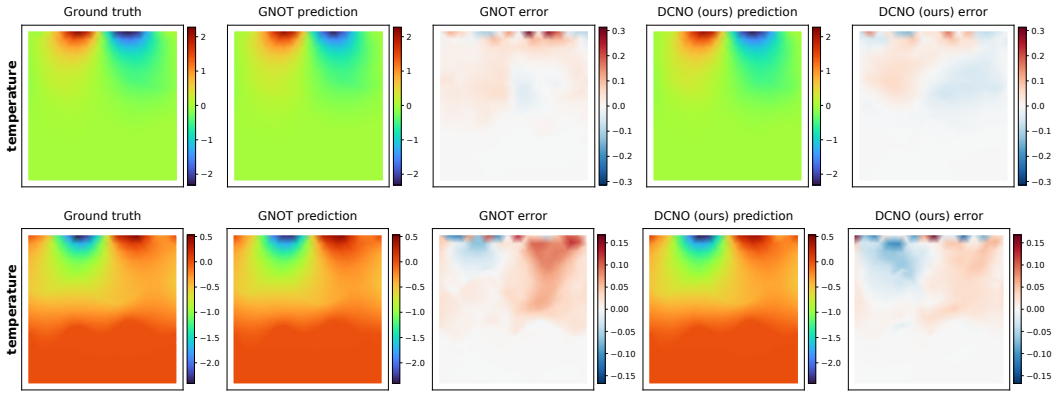


Figure 7: Additional qualitative results on Heat2D. Each row shows the reference temperature field, GNOT prediction and signed error, and DCNO prediction and signed error. Field and error scales are matched between methods within each row.

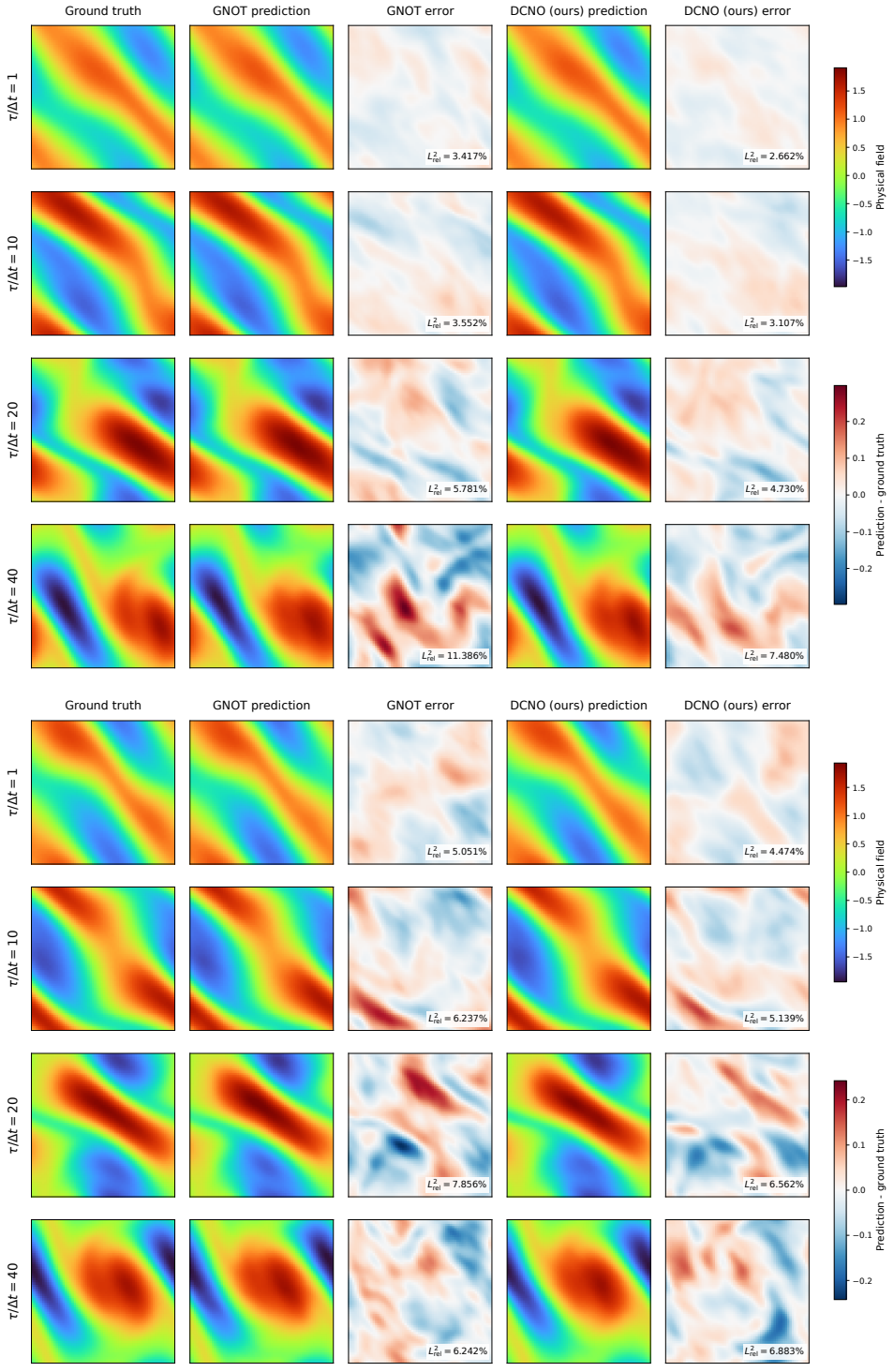


Figure 8: Additional qualitative results on NS2D across multiple forecast leads. The top and bottom panels show two representative trajectories. Within each panel, rows correspond to $\tau/\Delta t = 1, 10, 20, 40$, while columns show the ground-truth vorticity field, GNOT prediction and signed error, and DCNO prediction and signed error. Field and error color scales are shared across the rows within each panel, and the percentages shown in the error maps denote the relative L^2 errors.