Transferable, Time-Parallel Graph Dynamics via Space-Time Factorization

Wencong Yang **Chaopeng Shen*** **Abdolmehdi Behroozi**
Department of Civil and Environmental Engineering
Penn State University
University Park, PA 16802-1408, USA
wby5078@psu.edu, cxs1024@psu.edu, amb10399@psu.edu

Abstract

Time-parallel neural operators (FNO and successors) have transformed PDE surrogates on *regular* grids. Mesh-based operators (MeshGraphNets, Transolver) extend operator learning to *irregular* geometries, while jointly learning across diverse topologies and retaining non-recurrent temporal evaluation remains challenging. We propose **space-time factorization**: partition the model into local temporal operators (learned, graph-independent) and a spatial mixer that propagates information across nodes without graph-dependent learned parameters. This structural condition allows the same parameters to run on unseen graphs; non-recurrent instances also admit time-parallel evaluation. Accurate zero-shot transfer further depends on controlling three information leaks—graph-dependent weights, graph-specific features, and graph-correlated training distributions—and on suitable task physics and training coverage. The same protocol applies to two distinct spatial-mixer families tested here—fixed-spectral and learned-attention—suggesting the principle is not tied to a single mixer choice. We validate the principle in three domains. On simulating floodwave propagation in river networks, a 27K-parameter model trained on the Yangtze transfers zero-shot to the Mississippi with median NSE 0.998 in the boundary-conditioned operational evaluation, training $5.5\times$ faster than a stabilized recurrent baseline in a controlled one-epoch workload. On cross-geometry Navier–Stokes, the three-seed SepFNO mean relative vorticity error on five unseen targets falls by about 28% from 5 to 12 training geometries, with a complementary single-run SepTransolver trend. On cross-city traffic forecasting, zero-shot transfer reaches within 12.9% of an oracle trained on the target city. A consistent pattern emerges across these settings: cross-geometry error decreases with both the *number* of training geometries and their *relatedness* to the test, with the rate of accumulation depending on the spatial mixer. The result opens the path for pooled-data, foundation-style models for graph dynamics on irregular topologies.

1 Introduction

Operator learning on regular grids has become one of the dominant paradigms for fast Partial Differential Equations (PDE) surrogates: time-parallel models such as Fourier Neural Operators (FNO) report up to three orders-of-magnitude faster solves than traditional solvers [Li et al., 2021], and avoiding recurrent rollout also avoids the substantial activation-storage burden of backpropagation through time on long sequences [Gruslys et al., 2016]. Extending these benefits to irregular geometries—and, more broadly, to spatiotemporal prediction on graphs that arise in physical and engineered systems,

*Corresponding author: cxs1024@psu.edu.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

including flood propagation in river networks, fluid dynamics on unstructured meshes, and traffic forecasting on road networks—is an open problem. The workhorse architectures for these tasks—graph neural operators, MeshGraphNets, and spatiotemporal graph neural networks (GNNs)—couple message passing with recurrent time stepping: at each time step, a graph operator propagates information across nodes, and a Gated Recurrent Unit (GRU) or LSTM updates node states. This *entangled* design has an underappreciated consequence: it precludes access to time-parallel solvers. When a nonlinear graph operation sits inside the temporal recurrence (e.g., $h_{t+1} = \sigma(WAh_t + \cdots)$), the per-node temporal evolution is no longer linear time-invariant, the system can only be advanced sequentially, and standard tools from scientific computing—Fast Fourier Transform (FFT), spectral kernels, classical time-parallel integration algorithms (see §2)—are inapplicable. It is even unclear whether there is a viable path for using time-parallel algorithms to simulate complex nonlinear graph dynamics such as Navier–Stokes, since classical Parareal and MGRIT converge poorly on advection-dominated regimes. The same entanglement also degrades cross-graph transfer: even with shared weights [Hamilton et al., 2017, Gilmer et al., 2017], the learned dynamics become implicitly conditioned on training-graph topology when information leakage is present, limiting reliable transfer and pooled training across graphs.

We propose **space-time factorization** as the structural condition that removes both obstructions: partition the architecture into (a) *local temporal operators* whose learned parameters are graph-independent and act per node, and (b) *spatial mixers* that propagate information across nodes through fixed graph operators with no learned parameters or adjacency-free attention with shared learned projections. Two consequences follow: **efficiency** (parallel temporal solve without backpropagation through time, spatial mix in $O(K|E|)$ where K is the number of propagation steps and $|E|$ the number of graph edges) and **cross-graph transfer and knowledge accumulation** (no learned parameter touches the graph, so models deploy zero-shot by swapping the adjacency matrix, and data pooled from many topologies trains a single model whose accuracy can improve with broader training-geometry coverage, with gains depending on pool composition). The efficiency benefit is direct: once temporal and spatial operators are decoupled, the temporal step becomes per-node and linear-in-time, opening access to FFT, spectral kernels, time-parallel attention, and other non-recurrent solvers.

Our contributions: (1) **space-time factorization** as a design principle for graph dynamical systems, where local temporal operators are paired with fixed graph propagation; (2) a **systematic analysis** of three information leaks that break the factorization—graph-dependent weights, graph-dependent features, and graph-correlated training distributions—and the *time parallelism* and empirical pattern of *knowledge accumulation across training geometries* that the principle enables; (3) **validation** across three spatiotemporal domains plus a static one, with two negative controls that map out the applicability boundary.

2 Related Work

Neural operators on regular grids. The Fourier Neural Operator [Li et al., 2021] stacks FFT-based spectral kernels and predicts an entire output window in a single forward pass: the temporal solve is $O(T \log T)$, BPTT is avoided, and hundreds of time steps cost nearly the same as one [Gruslys et al., 2016]. Standard FNO assumes a regular grid; extending the same time-parallel benefits to irregular meshes is the open problem we target.

Operator learning on irregular geometries. Mesh-based and geometry-aware operator models—MeshGraphNets [Pfaff et al., 2021], Geo-FNO [Li et al., 2023a], GINO [Li et al., 2023b], AMG [Li et al., 2025], RIGNO [Mousavi et al., 2025], Sp^{2}GNO [Sarkar and Chakraborty, 2025], and Transolver [Wu et al., 2024]—extend operator learning to irregular domains, but two patterns recur: spatial operators carry mesh-dependent learned weights (relative-position edges, geometry-conditioned encoders) and time advances autoregressively, consuming substantial compute and memory on long horizons. Our work imposes graph-independent learned weights and a time-parallel temporal solver simultaneously; Appendix H compares cross-geometry benchmarks (PDEBench, SURF, MGN cross-geometry studies).

Classical time-parallel methods. Parareal [Lions et al., 2001, Gander and Vandewalle, 2007] and MGRIT [Falgout et al., 2014] have well-established convergence on linear, diffusion-dominated PDEs but converge slowly and non-robustly on nonlinear advection-dominated problems including Navier–Stokes [De Sterck et al., 2020]. Operator-splitting (Chorin–Temam) decouples partially but

still advances time sequentially. FNO sidesteps both by *learning* per-layer-linear spectral kernels stacked with nonlinear activations—on regular grids. Whether this trade-off transfers to irregular meshes with graph-free spatial mixing is the empirical question our PDE experiments answer.

Spatiotemporal graph neural networks. DCRNN [Li et al., 2018], STGCN [Yu et al., 2018], Graph WaveNet [Wu et al., 2019b], and AGCRN [Bai et al., 2020] all interleave learned dynamics with the graph: STGCN combines fixed-adjacency spatial graph convolutions with learnable temporal convolution blocks; Graph WaveNet learns adaptive adjacency; DCRNN places diffusion *inside* the GRU update; AGCRN ties node-specific embeddings into the recurrence. These choices affect temporal parallelism and transfer in different ways: recurrence limits time parallelism, while source-node-specific embeddings directly obstruct transfer. Our SepGRU keeps the GRU temporal core (appropriate for the short prediction horizon of standard traffic benchmarks) but moves graph propagation outside the recurrence with feature-conditioned, graph-free hop weights—demonstrating that the factorization principle applies even when FFT-in-time is impractical.

Inductive and decoupled graph learning. Inductive methods—MPNNs [Gilmer et al., 2017], GraphSAGE [Hamilton et al., 2017], GCNs [Kipf and Welling, 2017]—share parameters across nodes and graphs but are evaluated almost exclusively on static problems. SGC [Wu et al., 2019a], SIGN [Frasca et al., 2020], and APPNP [Klicpera et al., 2019] decouple fixed propagation from graph-free predictors, with graphon analyses studying filter transferability [Ruiz et al., 2021]. Extending this to nonlinear temporal dynamics is non-trivial: temporal evolution opens leakage channels (magnitude distributions, boundary-feature memorization) that have no analogue in static node classification, and time-parallelism becomes available only when per-step linearity is enforced layer-by-layer.

3 Method: Space-Time Factorization

3.1 Problem Setting

Consider a spatiotemporal prediction task on a graph $G = (V, E)$ with adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$. Each node i has local features $\mathbf{x}_i \in \mathbb{R}^d$ and a time-varying target $\mathbf{y}_i(t)$. We want a model f_θ trained on graphs $\{G_1, \ldots, G_K\}$ that generalizes to an unseen graph G_{test} with different N, different $\mathbf{A}$, but the same type of node features, dynamics, and prediction task.

3.2 The Factorization Principle

Standard spatiotemporal GNNs interleave learned message passing with recurrent time stepping: at each time t, they compute hidden states $\mathbf{h}(t) = \text{GRU}(\mathbf{h}(t-1),\ \text{MP}(\mathbf{h}(t-1), \mathbf{A}))$, where GRU is a Gated Recurrent Unit [Cho et al., 2014] (any recurrent cell with learned dynamics works equally well for our argument), MP is graph message passing, and $\mathbf{h}(t) \in \mathbb{R}^{N \times d}$ is the snapshot of d-dimensional hidden states across all N nodes at time t (not a time series). This update couples learned parameters θ with the adjacency $\mathbf{A}$ at every step. We propose factorizing these:

> **Space-Time Factorization:** Partition the model into (a) *local temporal operators* with learned parameters θ acting on each node independently, and (b) a *spatial mixer* $\mathcal{M}$ that propagates information across nodes using *no graph-dependent learned parameters*. Temporal and spatial operations compose but do not interleave.

Figure 1 contrasts graph mixing inside a recurrent update with our factorized window-based template.

Formally, the forward pass has the structure:

$$\mathbf{h}_i^{(\ell+1)}(t) = f_\theta\Big(\mathbf{x}_i(t),\ \big[\mathcal{M}(\mathbf{h}^{(\ell)}(t); \mathbf{A})\big]_i\Big) \tag{1}$$

where the spatial mixer $\mathcal{M}$ propagates information across nodes *without learned parameters that condition on* $\mathbf{A}$. This admits two construction patterns: (a) a **fixed graph operator** $\mathcal{M}(\mathbf{h}; \mathbf{A}) = G(\mathbf{A})\mathbf{h}$ where $G(\mathbf{A}) \in \{\mathbf{A}, \mathbf{A}^k, \sum_k \alpha_k \mathbf{A}^k, T_k(\mathbf{A})\}$ denotes fixed one-hop propagation, k-hop propagation, weighted multi-hop diffusion, or Chebyshev polynomial graph filtering, respectively, with no graph-specific learned parameters (Our SepFNO, SepGFNO, SepGRU below); the rich local-feature information enters via $\mathbf{h}$, while $G(\mathbf{A})$ propagates those feature-encoded states through the fixed adjacency. Or (b) a **learned cross-node operator** (e.g., self-attention or Physics-Attention slice

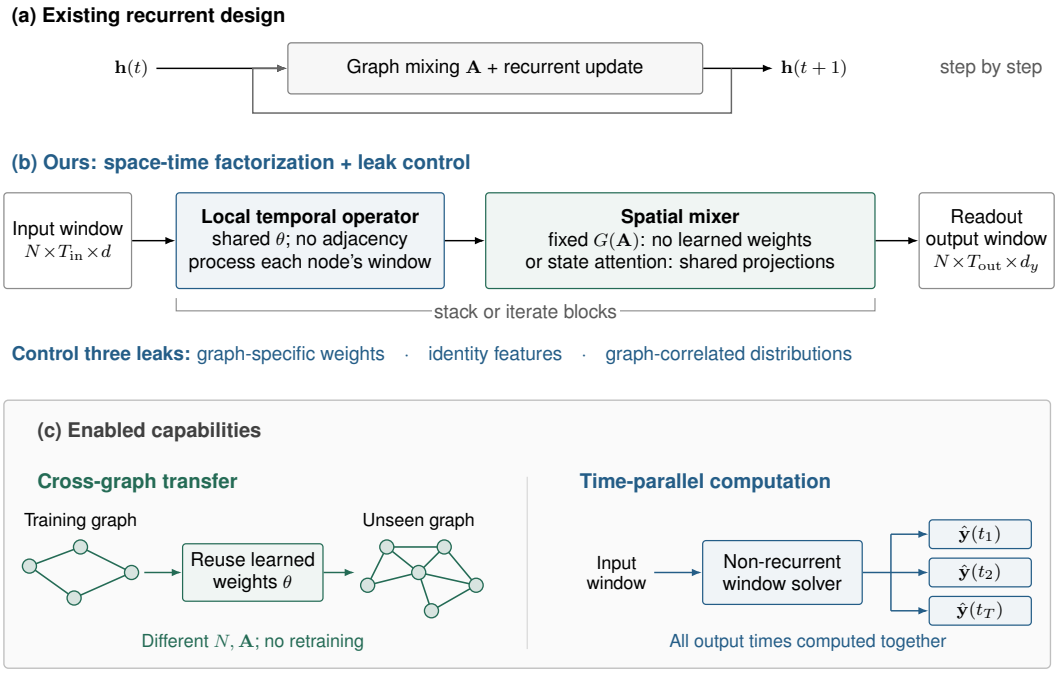

Figure 1: Method schematic: space-time factorization and its enabled capabilities. N is the node count, $\mathbf{A}$ the adjacency matrix, θ the shared learned parameters, and $T_{\text{in}}/T_{\text{out}}$ the input/output window lengths.

tokens in our SepTransolver below) whose learned weights (Q, K, V projections, slice projection W) act on *node states* but never reference $\mathbf{A}$—node-to-node coupling emerges from softmax similarities of the states themselves. Both patterns satisfy the constraint that no learned parameter encodes graph identity (in pattern (b), the mesh may still be used during preprocessing to compute features such as distance-to-boundary, but is never consulted inside any layer). f_θ is a shared temporal operator (MLP, FFT kernel, or GRU) applied identically to every node.

Iterating for spatial reach and nonlinearity. Equation (1) shows a single application of $\mathcal{M}$; practical instantiations stack or iterate this operation. For fixed graph operators, stacking can be done by repeated propagation, higher-order graph polynomials, or multiple operator layers. For learned graph-free cross-node operators, multiple attention layers can be stacked. What the factorization forbids is not iteration but *graph-conditioned entanglement*: spatial mixing must not be interleaved with learned parameters that depend on $\mathbf{A}$. The composition $f_\theta \circ \mathcal{M} \circ f_\theta \circ \mathcal{M} \circ \cdots$ preserves graph-independence layer-by-layer while recovering nonlinearity through stacking. This extends the SGC [Wu et al., 2019a] / SIGN [Frasca et al., 2020] decoupling logic from static classification to spatiotemporal prediction, enabling efficiency and knowledge-accumulation benefits at the cost of additional leak control (next subsection).

3.3 Information Leaks: Three Levels of Graph Dependence

Standard inductive GNNs already share weights across nodes, but graph-specific information can still leak into the learned representation through weights, features, and training distributions. We therefore impose three design constraints.

Level 1: Graph-free weights. All learned parameters must be independent of the graph. Per-node learnable embeddings, Laplacian positional encodings [Dwivedi et al., 2020], and learned adjacency matrices violate this; SGC [Wu et al., 2019a], SIGN [Frasca et al., 2020], GraphSAGE [Hamilton et al., 2017], and our SepFNO/SepGRU/SepGFNO satisfy it.

Table 1: Shared-parameter instantiations. Cross-graph deployment is defined for all five; time parallelism applies only to the non-recurrent dynamics.

Model	Shared temporal component	Spatial mixer	Time-parallel?
SepGFNO	MC coefficients, FFT-IIR	fixed upstream propagation	Yes
SepFNO	temporal FFT kernels	fixed Chebyshev polynomials	Yes
SepTransolver	windowed input/output	learned state attention	Yes
SepGRU	shared GRU	fixed adjacency powers	No
SepHybrid	static MLP	fixed pre-smoothing	N/A (static)

Level 2: Graph-free features. Input features must describe transferable local properties rather than graph-specific identity. Absolute or normalized coordinates are common leaks; local descriptors with consistent meaning across graphs (distance to boundary, curvature, slope) are acceptable.

Level 3: Graph-correlated training distributions. Even with graph-free weights and features, the training distribution can carry graph-specific signatures (e.g., characteristic magnitude ranges per geometry); a model can then infer graph identity from data statistics. Distribution augmentation—rotation, magnitude scaling, feature dropout—is the standard remedy for breaking these correlations. The remedy is architecture-dependent: magnitude perturbation helps fixed-polynomial mixers but can hurt softmax-based attention whose weights are activation-scale sensitive (§4.2).

3.4 Per-Step Linearity as the Enabler of Time-Parallelism

We sidestep classical Parareal/MGRIT difficulties (§2) by following the FNO trade-off: each temporal kernel is FFT-able through per-step linearity, while stacking with nonlinear activations recovers nonlinear input–output behavior. Our SepTransolver instead achieves time-parallelism through windowed channel I/O (non-recurrent single forward pass, $O(T)$ scaling vs SepFNO's $O(T \log T)$). Strict factorization cannot model state-dependent spatial coupling without more layers or hybrid extensions; the tradeoff is favorable when coupling is primarily local-attribute-dependent.

3.5 Necessary Conditions for Applicability

Because no learned parameter may depend on a particular graph, two task conditions matter: (i) connectivity can be separated from the governing physics (when edge weights themselves carry equation-defining coefficients, as with the admittance matrix in power flow, this separation fails); and (ii) features carry local physics in transferable form—reach attributes for rivers, boundary descriptors for irregular-mesh PDEs, and sensor statistics for traffic. These conditions do not guarantee accuracy; leak control (§3.3), training coverage, and backbone capacity also matter. A non-recurrent temporal solver is an additional requirement for time parallelism (§3.4). The negative controls of §4.4 test conditions (i) and (ii): power flow violates (i), airport networks violate (ii).

3.6 Instantiations Across Domains

All instantiations share a template: a shared encoder maps local attributes to states, graph-independent heads produce coefficients, a shared temporal processor evolves each node, and a spatial mixer couples nodes. Their temporal solvers and spatial mixers differ (Table 1; further details in Appendix A). SepFNO uses fixed Chebyshev polynomials, while SepTransolver uses adjacency-free learned Physics-Attention within this template.

River routing (SepGFNO, known physics). The task estimates the propagation of floodwaves downstream across thousands of time steps through a river network—a critical procedure for issuing flood warnings. The Muskingum–Cunge (MC) method solves a simplified form of the Saint-Venant continuity and momentum equations over large open-channel networks across many time steps. A per-reach MLP maps local attributes (slope, roughness, upstream area) to MC coefficients of the linear recurrence [Bindas et al., 2024]: $Q_i(t) = c_1 I_i(t) + c_2 I_i(t-1) + c_3 Q_i(t-1) + c_4 q'_i(t)$. Spatial propagation iterates the fixed upstream-connectivity matrix $\mathbf{N}$ for D rounds (network diameter; Appendix E.1).

PDE solving (SepFNO and SepTransolver). The task is solving 2D incompressible Navier–Stokes (vorticity formulation) on irregular meshes—the classical operator-learning benchmark—and zero-shot transferring to a held-out geometry. Each mesh node carries local boundary features (distance to boundary, surface normal, curvature) and *no absolute coordinates*. SepFNO applies per-frequency FNO-style spectral kernels and mixes neighboring information via fixed Chebyshev polynomials $T_0(\mathbf{A}), \ldots, T_K(\mathbf{A})$. SepTransolver replaces the Chebyshev mixer with Physics-Attention slice tokens (learned weights that do not reference $\mathbf{A}$) and uses windowed channel I/O instead of FFT. Both are time-parallel under our protocol (Appendices E.2, G).

Traffic forecasting (SepGRU, learned dynamics). The task is short-horizon traffic-speed forecasting on road sensor networks (the standard DCRNN benchmark) and zero-shot transferring across cities with disjoint sensor IDs. Each sensor has local statistics (mean speed, variance, congestion fraction). A shared GRU encodes each sensor's series independently; an MLP maps local statistics to per-node hop weights $\boldsymbol{\alpha}_i$ for fixed multi-hop diffusion $\sum_k \alpha_{i,k}(\mathbf{A}^k\mathbf{h})_i$. The graph enters only through precomputed adjacency powers (Appendix E.3).

Citation classification (SepHybrid, static graphs, breadth check). SIGN-style pre-smoothing followed by graph-free MLP heads (Appendix I) to check if our principle extends to static graphs.

4 Experiments

We evaluate whether graph-independent learned components, fixed graph operators, and leakage control improve zero-shot transfer across structurally different graphs. The experiments cover the three spatiotemporal domains and the static breadth-check domain introduced in §3.6, plus two negative controls. In domains where the temporal dynamics admit convolutional or physics-structured solvers, the same separation also provides computational advantages.

4.1 River Flood Routing

A single SepGFNO model trained on bounded-diameter Yangtze subgraphs ($D_{\max} = 50$, 50,859 reaches) is deployed zero-shot on the Mississippi basin with no basin-specific adaptation, testing whether a physics-anchored, locally parameterized routing model transfers across a substantially different network while remaining scalable to basin-wide inference. The target is simulated reference discharge with known lateral inflows and reach attributes.

Table 2: Yangtze→Mississippi operational evaluation on the common 86,133-reach cohort. Peak error is absolute percent; timing is absolute hours. Boundary flows at trunk injections use reference discharge; metrics cover 2,400 hourly steps per reach, not a calendar year.

Model	NSE med.	KGE med.	Peak (%)	Timing (h)	NSE P10
SepGFNO, trained	**0.998**	**0.983**	**1.51**	**1**	**0.985**
SepGFNO, anchor-only	0.569	0.292	50.01	1	-1.118
GraphSAGE+GRU, ep150	0.715	0.525	32.77	12	-5.800

SepGFNO transfers with little loss across geographically and topologically distinct regions (Table 2). The Yangtze-trained model reaches median NSE 0.998 and KGE 0.983 on the Mississippi operational evaluation; its lower-decile NSE is 0.985. The trained model's median full-flow and high-flow relative L_2 errors are 0.0280 and 0.0240, versus 0.5011 and 0.5045 for the anchor-only control (Appendix C). Thus the physical anchors alone do not account for the learned model's magnitude and high-flow accuracy. Both anchored variants have 1 h median peak-timing error, so timing skill cannot be attributed entirely to learning. SepGFNO learns a basin-independent mapping from local reach attributes (slope, length, roughness, hydraulic geometry, upstream area) to routing coefficients; topology enters through the fixed propagation matrix $\mathbf{N}$. The reach-wise maps and drainage-area summaries in Appendix Figure S1 show that strong transfer spans headwaters and mainstem reaches, with a somewhat wider error spread in intermediate tributaries.

Baselines and efficiency. The stabilized sparse GraphSAGE+GRU baseline supports basin-wide evaluation but has lower transfer accuracy (Table 2). In a controlled one-epoch workload on a Tesla

V100S-PCIE-32GB, SepGFNO FFT-IIR takes 72 s and 2.5 GB peak allocated GPU memory, versus 3,121 s and 5.4 GB for the same architecture with sequential IIR, and 396 s and 24.2 GB for stabilized GraphSAGE+GRU (Appendix F). The FFT version is $43.3\times$ faster than its sequential control and $5.5\times$ faster than GraphSAGE+GRU for this workload; these are not matched-accuracy convergence comparisons.

For the river setting we use a closed-form Muskingum–Cunge spectral solver (§3) rather than an L-layer FNO stack: the recurrence is known and exact under the linearity assumption, so a learned spectral kernel would only discard this physical prior. The L-layer design is appropriate for PDE solving, where the operator is not known in closed form. SepTransolver-style learned attention is in principle applicable but practically infeasible here: $T \approx 10^3$ hourly steps and sparse tree adjacency over 5×10^4 reaches would force either many autoregressive windows or prohibitive memory; Physics-Attention's slice tokens (typically 32) cannot meaningfully summarize 5×10^4 reaches spanning headwaters, mainstem, and tributaries. The efficiency advantage of time-parallel solvers also scales proportionally to the number of time steps. Such computational efficiency helps enable continental- and global-scale streamflow modeling across all river reaches, supporting the expanding scope of physics-embedded hydrologic models [Song et al., 2025, Ji et al., 2025, Yang et al., 2026].

4.2 PDE Solving on Irregular Geometries

SepFNO, our first Navier–Stokes solver, is trained across fundamentally different, non-diffeomorphic geometries ($\sim$1,200–1,700 mesh nodes; Appendix J) and evaluated zero-shot on a held-out geometry. While operator learning on irregular geometries is an active area with related benchmarks (PDEBench [Takamoto et al., 2022], SURF [Künzli et al., 2024], MGN cross-geometry studies [Schmöcker et al., 2024]), we are not aware of a standardized benchmark that specifically tests how cross-geometry error scales with the number and diversity of training geometries (Appendix H surveys these). Our second solver adapts Transolver's Physics-Attention [Wu et al., 2024] as the spatial mixer within our protocol, yielding **SepTransolver** (Appendix G). Canonical Transolver supports learning across diverse geometries but does not explicitly target zero-shot cross-geometry transfer from single-geometry training; SepTransolver is our construction with five protocol changes (time-windowed channel I/O, graph-free boundary features, multi-mesh shared-weight batching, removal of scalar-time conditioning, stability-tuned LR; Appendix G). We also adapt MeshGraphNets [Pfaff et al., 2021] as an external mesh-based baseline. We additionally construct three ablation baselines—**MLP_FNO**, **GCN_FNO**, and **GraphSAGE_FNO**—by replacing SepFNO's spatial Chebyshev mixer with, respectively, no spatial coupling, a learned GCN aggregator [Kipf and Welling, 2017], and a GraphSAGE 1-hop aggregator [Hamilton et al., 2017]. MLP_FNO and GraphSAGE_FNO retain temporal FFT processing; GCN_FNO uses per-time-step graph convolutions. These baselines compare alternative spatial processing.

Table 3: Original triangle+notch cross-geometry NSE at 4g, matched-aug protocol (Appendix J); separate from the five-target scaling sweep. *External*: published methods adapted (Appendix H). *Ablation*: variants replacing SepFNO's spatial mixer. SepTransolver no-aug reaches 0.046.

Method	Params	L_2
SepFNO (ours)	**454K**	**0.041**
External (adapted)		
SepTransolver	630K	0.045
MeshGraphNet	419K	0.248
Ablation (ours)		
GraphSAGE_FNO	307K	0.060
MLP_FNO	196K	0.110
GCN_FNO	48K	0.279

The PDE results have two complementary parts. We first measure how each leak-closing step from §3 contributes with 4 training geometries (4g, Table 4); this isolates per-component effects at a fixed protocol where all baselines train stably. We then examine how cross-geometry error evolves as more training geometries are added (Figure 3), testing whether the principle supports *knowledge accumulation* across geometries.

Effect of leak control and architectural choices. Closing the Level-2 (coordinate) and Level-3 (distribution) leaks of §3, together with the Chebyshev mixer and a longer training schedule, more than halves the cross-geometry error from L_2=0.092 down to 0.041 at four training geometries, bringing it within $\sim2\times$ of the test-only oracle (Table 4). Each step contributes measurably and no single intervention dominates: absolute coordinates are the largest single leak (removing (x, y) reduces error by 27%), the Chebyshev series adds an additional 10%, and the augmentations combined

remove a further 26% by reducing geometry-specific signatures that persist in boundary features and vorticity magnitudes after coordinate identity is removed. These component-level effects explain the strength of the benchmarks reported below.

Table 4: PDE ablation at 4g: each row modifies the previous configuration.

Configuration	Cross-geo L_2	Main effect
SepFNO + position (x, y)	0.092	coordinate/geometry identity leak
− remove position	0.068 (-27%)	absolute-coordinate memorization
+ Chebyshev $T_0 \dots T_6(\mathbf{A})$	0.061 (-10%)	richer fixed spatial mixing
+ rotation aug + BF dropout	0.056 (-8%)	boundary-shape memorization
+ ω-scale augmentation	0.046 (-18%)	magnitude–geometry correlation
+ longer schedule (300ep+ft)	**0.041** (-11%)	convergence/refinement
Oracle (test-only, ref.)	$\sim$0.018	—

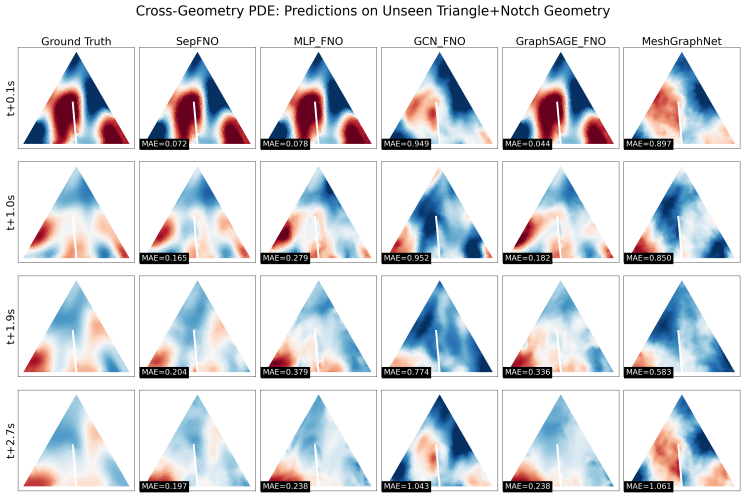

Figure 2: Representative 20-step vorticity rollout on an unseen triangle+notch geometry. SepFNO preserves the main vorticity structures, while weaker baselines produce visibly distorted fields.

Figure 2 illustrates the quantitative comparison with a 20-step vorticity rollout on the held-out triangle-with-notch mesh: SepFNO retains the principal flow structures while weaker baselines visibly distort them.

Cross-architecture comparison (Table 3). Both SepFNO (0.041) and SepTransolver (0.045) outperform all structural-ablation baselines by 1.3–7×, and their agreement to within $\sim$10% provides empirical evidence that the factorization principle accommodates more than one spatial-mixer family (fixed-spectral and learned-attention here). The remaining baselines fail in ways predicted by the leakage framework above: **GCN_FNO** (0.279) couples fixed-adjacency aggregation with learned feature transformations, potentially overfitting to graph-specific patterns; **MeshGraphNet** (0.248) uses relative-position edge features that are standard in single-mesh settings but encode geometry identity under cross-geometry transfer (a Level-2 leak embedded in the spatial mixer itself); **GraphSAGE_FNO** (0.060) uses 1-hop mean aggregation that satisfies the leakage constraints but is expressively weaker than the Chebyshev $K{=}6$ mixer; **MLP_FNO** (0.110) has no learned cross-node mixing, although its derivative inputs contain local spatial information. The MLP_FNO vs. GCN_FNO/MeshGraphNet comparison surfaces a non-obvious lesson: *mis-specified learned spatial coupling is worse for cross-geometry transfer than limited, fixed spatial coupling.*

Knowledge accumulation: quantity and relatedness. Figure 3 tests *cross-geometry knowledge accumulation across two space-time-factorized model families*: fixed-spectral SepFNO and learned-attention SepTransolver. On the same five unseen targets, the final three-seed SepFNO sweep reduces mean relative vorticity error from 0.0923 ± 0.0037 at 5g to 0.0663 ± 0.0005 at 12g (about 28%);

a separate single-run SepTransolver sweep falls from 0.1013 at 6g to 0.0652 at 15g (Appendix Table S8). The common direction across different spatial mixers supports accumulation beyond one architecture or one test geometry. It is an empirical trend across sampled pools, not a monotone law: SepFNO worsens at 6g and improves little from 9g to 12g. Equal-size SepTransolver controls also show that pool composition matters: adding a sinusoidal channel improves mean error from 0.0994 to 0.0754, whereas adding an hourglass worsens it to 0.1102. Matched target-only oracle gaps still vary by geometry (Appendix J).

The six-target check in Appendix Table S9 uses fixed 4g checkpoints without target adaptation. SepFNO has the lowest vorticity and reconstructed-velocity L_2 on every held-out geometry among the five benchmarked models; its across-target means are $0.0482/0.0283$, versus $0.0674/0.0398$ for the next-best GraphSAGE_FNO. Pressure is absent from this vorticity formulation. Per-node analysis finds larger boundary errors (Appendix D). As a public-data test beyond our generated geometries, we also train SepFNO on the unsteady Schmöcker cylinder-flow benchmark [Schmöcker et al., 2024] and transfer zero-shot to the unseen stretched cylinder. In the matched laminar subset ($Re < 47$, $n = 11$), its median 20-step velocity RMSE is 76.0×10^{-3}, versus 163.0×10^{-3} for native MeshGraphNet (Appendix Table S10). This supports transfer in a separate geometry family, although the available MeshGraphNet checkpoint is undertrained; the broader Reynolds-number results and this fairness limit are given in the appendix.

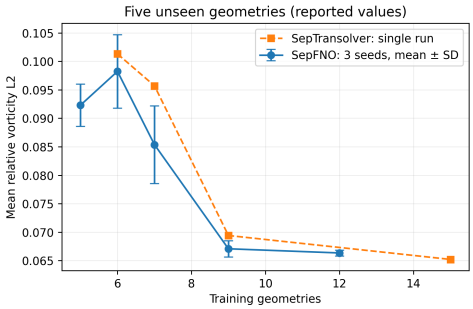

Figure 3: Cross-geometry knowledge accumulation across two space-time-factorized families on five unseen geometries. SepFNO (fixed-spectral mixer) shows three-seed mean $\pm$ SD; SepTransolver (learned-attention mixer) is single-run. Pools differ by size and composition.

4.3 Traffic Speed Forecasting

Traffic forecasting tests a learned-dynamics setting where no physical recurrence is available. We use the DCRNN benchmark protocol [Li et al., 2018]: models are trained on METR-LA (207 sensors) and deployed zero-shot on PEMS-BAY (325 sensors). Unlike the river and PDE cases, the protocol uses a short prediction horizon (12 steps in, 12 steps out), where FFT-in-time offers little wall-clock advantage over a recurrent core; temporal dynamics are therefore learned by a shared GRU, while the factorization principle is enforced at the spatial side. Spatial coupling is introduced through fixed multi-hop diffusion over the target road graph, with node-wise hop weights predicted from local sensor statistics rather than from sensor identities. The choice of temporal solver (FFT versus GRU) is domain-driven: the principle constrains where graph topology may enter, not which temporal solver the local dynamics use. At deployment, SepGRU-v2 computes local hop weights from unlabeled target-city sensor statistics; no target-city labels are used for training or adaptation.

Table 5: Traffic METR-LA$\rightarrow$PEMS-BAY zero-shot transfer (DCRNN 12-step protocol; 30-min MAE on PEMS-BAY). *NodeEmbedGRU*/*EntangledGRU* are our re-implementations of the AGCRN [Bai et al., 2020]/DCRNN [Li et al., 2018] cross-city paradigms. SepGRU-v2 robustness: 2.16 ± 0.01 over 3 seeds.

Model	Transfer MAE	Oracle gap	Transfer factor tested
SepGRU-v2 + dropout (ours)	**2.13**	**+12.9%**	feature-conditioned diffusion + regularization
SepGRU-v2 (ours)	2.16	+14.3%	feature-conditioned diffusion weights
EntangledGRU (DCRNN-style)	2.18	+16.8%	shared graph-in-recurrence, no node IDs
SepGRU-v1 (ours, no learned α)	2.23	+19.3%	fixed hop weights, no local adaptation
NodeEmbedGRU (AGCRN-style)	2.91	+55.9%	source-node embeddings

The traffic results isolate a Level-1 leak. NodeEmbedGRU follows standard practice [Bai et al., 2020] in assigning each sensor a learnable embedding vector indexed by sensor ID; these embeddings are graph-specific by construction, so the embedding for "LA sensor 47" is meaningless on PEMS-BAY

and the oracle gap rises to 55.9%. Replacing them with diffusion weights predicted from *local sensor statistics* (mean speed, variance, congestion fraction) restores transferability: the gap drops to 14.3% for SepGRU-v2 and 12.9% with dropout in encoder and coefficient head. SepGRU-v1, with fixed diffusion and no per-node adaptation, is worse ($+19.3\%$), so some local adaptation helps—as long as it is predicted from transferable features rather than learned per identity. EntangledGRU, which places graph diffusion inside the recurrent update without node-specific weights, remains close to SepGRU-v2 (2.18 vs. 2.16): cross-city transfer breaks most clearly when learned parameters are tied to source sensors, not from graph use inside the recurrence per se.

4.4 Breadth Check and Negative Controls

Citation networks (breadth check beyond spatiotemporal): on GDABench [Liu et al., 2024] (DBLP, ACM, Citation; Appendix I), a SIGN-style pre-smoothed model with graph-free MLP heads reaches 64.8% average Macro-F1 vs. 63.7% for a 3-layer GCN across six source→target pairs—a modest gain confirming the rule applies beyond temporal dynamics. As negative controls, **Power flow** (IEEE 24/39/118-bus, cross-system transfer): the admittance matrix encodes line impedances that enter Kirchhoff's laws directly—the graph *is* the physics. Graph models incur $+500\%$ transfer gaps; a graph-free MLP on local injection features increases error by 12%. **Airport routes** (GDABench Brazil/Europe/USA): node features are one-hot degree encodings (graph-derived, not physical), so SepGNN (44.3% F1) falls below MLP (54.5%) and GCN (50.2%) (applicability checklist in Appendix B).

5 Conclusion

We have argued that separating temporal evolution from graph propagation changes what kinds of generalization and scaling become possible for graph-based dynamical systems. Once topology no longer participates in the recurrent update, time-parallel solvers become applicable on irregular graphs—a regime classical time iteration struggles with. The contribution is the **factorization protocol** (graph-free features, time-windowed I/O, fixed or graph-free spatial mixing, architecture-tuned leak control), not a new primitive layer. Empirically (Fig. 3), the same protocol unlocks both fixed-spectral (SepFNO/Chebyshev) and learned-attention (SepTransolver/Physics-Attention) mixers, with cross-geometry error decreasing in both training-geometry quantity and test-relatedness. The three domains exemplify distinct spatial-mixer choices appropriate to the regime—MC/FFT-IIR with fixed upstream propagation for rivers, Chebyshev or Physics-Attention mixing for PDEs, and GRU dynamics with fixed multi-hop diffusion for traffic.

Several limitations remain. The PDE cross-geometry gap is reduced by data scaling but not yet closed, and a linear temporal kernel cannot capture state-dependent spatial coupling without additional regimes or layer stacking; physics-informed extensions—such as embedding a differentiable Poisson solver or imposing soft sensitivity constraints around boundaries—are a natural next step.

Two directions stand out: scaling laws for graph dynamical systems—how cross-graph error decreases with the number, diversity, and similarity of training graphs—and opening a path toward foundation models for physical simulation, where many graph instances are pooled and a single model generalizes across them. We hope this paper provides one structural argument for why such an objective is reachable, and a concrete starting point for it.

Funding

This work is supported by the Cooperative Institute for Research to Operations in Hydrology (CIROH) through the National Oceanic and Atmospheric Administration (NOAA) Cooperative Agreement (Grant NA22NWS4320003) and National Aeronautics and Space Administration (NASA) grant 80NSSC24K1646.

Competing Interests

C.S. has financial interests in HydroSapient, Inc., a company which could potentially benefit from the results of this research. These interests have been reviewed by The Pennsylvania State University in

accordance with its individual conflict of interest policy, for the purpose of maintaining the objectivity and the integrity of research at The Pennsylvania State University.

References

Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. In *Advances in Neural Information Processing Systems*, volume 33, pages 17804–17815, 2020.

Tadd Bindas, Wen-Ping Tsai, Jiangtao Liu, Farshid Rahmani, Dapeng Feng, Yuchen Bian, Kathryn Lawson, and Chaopeng Shen. Improving river routing using a differentiable Muskingum–Cunge model and physics-informed machine learning. *Water Resources Research*, 60(1):e2023WR035337, 2024.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*, 2014.

Hans De Sterck, Stephanie Friedhoff, Alexander J. M. Howse, and Scott P. MacLachlan. Convergence analysis for parallel-in-time solution of hyperbolic systems. *Numerical Linear Algebra with Applications*, 27(1):e2271, 2020.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. 2020.

Robert D. Falgout, Stephanie Friedhoff, Tzanio V. Kolev, Scott P. MacLachlan, and Jacob B. Schroder. Parallel time integration with multigrid. *SIAM Journal on Scientific Computing*, 36(6):C635–C661, 2014.

Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. SIGN: Scalable inception graph neural networks. In *ICML Workshop on Graph Representation Learning*, 2020.

Martin J. Gander and Stefan Vandewalle. Analysis of the parareal time-parallel time-integration method. *SIAM Journal on Scientific Computing*, 29(2):556–578, 2007.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272, 2017.

Audrunas Gruslys, Remi Munos, Ivo Danihelka, Marc Lanctot, and Alex Graves. Memory-efficient backpropagation through time. In *Advances in Neural Information Processing Systems*, 2016.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Haoyu Ji, Yalan Song, Tadd Bindas, Chaopeng Shen, Yuan Yang, Ming Pan, Jiangtao Liu, Farshid Rahmani, Ather Abbas, Hylke Beck, Kathryn Lawson, and Yoshihide Wada. Distinct hydrologic response patterns and trends worldwide revealed by physics-embedded learning. *Nature Communications*, 16:9169, 2025.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2019.

Stefan Künzli, Florian Grötschla, Joël Mathys, and Roger Wattenhofer. SURF: A generalization benchmark for GNNs predicting fluid dynamics. In *Learning on Graphs Conference (LoG)*, 2024.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

Zhihao Li, Haoze Song, Di Xiao, Zhilu Lai, and Wei Wang. Harnessing scale and physics: A multi-graph neural operator framework for pdes on arbitrary geometries. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 729–740, 2025. doi: 10.1145/3690624.3709173.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

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, 2023a.

Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-informed neural operator for large-scale 3d pdes. In *Advances in Neural Information Processing Systems*, 2023b.

Jacques-Louis Lions, Yvon Maday, and Gabriel Turinici. A "parareal" in time discretization of PDE's. *Comptes Rendus de l'Académie des Sciences–Series I–Mathematics*, 332(7):661–668, 2001.

Meihan Liu, Zhen Zhang, Jiachen Tang, Jiajun Bu, Bingsheng He, and Sheng Zhou. Revisiting, benchmarking and understanding unsupervised graph domain adaptation. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks)*, 2024.

Sepehr Mousavi, Shizheng Wen, Levi Lingsch, Maximilian Herde, Bogdan Raonić, and Siddhartha Mishra. RIGNO: A graph-based framework for robust and accurate operator learning for PDEs on arbitrary domains. *arXiv preprint arXiv:2501.19205*, 2025.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

Luana Ruiz, Fernando Gama, and Alejandro Ribeiro. Graph neural networks: Architectures, stability, and transferability. *Proceedings of the IEEE*, 109(5):660–682, 2021.

Subhankar Sarkar and Souvik Chakraborty. Spatio-spectral graph neural operator for solving computational mechanics problems on irregular domain and unstructured grid. *Computer Methods in Applied Mechanics and Engineering*, 2025.

Robin Schmöcker, Alexander Henkes, Julian Roth, and Thomas Wick. Generalization capabilities of MeshGraphNets to unseen geometries for fluid dynamics. *arXiv preprint arXiv:2408.06101*, 2024.

Yalan Song, Tadd Bindas, Chaopeng Shen, Haoyu Ji, Wouter J. M. Knoben, Leo Lonzarich, Martyn P. Clark, Jiangtao Liu, Katie van Werkhoven, Sam Lamont, Matthew Denno, Ming Pan, Yuan Yang, Jeremy Rapp, Mukesh Kumar, Farshid Rahmani, Cyril Thébault, Richard Adkins, James Halgren, Trupesh Patel, Arpita Patel, Kamlesh Arun Sawadekar, and Kathryn Lawson. High-resolution national-scale water modeling is enhanced by multiscale differentiable physics-informed machine learning. *Water Resources Research*, 61(4):e2024WR038928, 2025. doi: 10.1029/2024WR038928.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An extensive benchmark for scientific machine learning. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, pages 6861–6871, 2019a.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. In *International Conference on Machine Learning*, 2024.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *International Joint Conference on Artificial Intelligence*, 2019b.

Wencong Yang, Leo Lonzarich, Yalan Song, Haoyu Ji, Ming Pan, Kathryn Lawson, and Chaopeng Shen. Hourly U.S.-wide flood simulation beyond the limits of traditional and data-driven models. *arXiv preprint arXiv:2609.06794*, 2026. doi: 10.48550/arXiv.2609.06794.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *International Joint Conference on Artificial Intelligence*, 2018.

A Instantiations of the Factorization Principle

Table S1 summarizes how the factorization template is instantiated across the five models in the paper. The spatial mixer is the modular slot whose choice depends on the regime (Long horizons and large sparse graphs favor fixed propagation for scalability; smaller irregular meshes can afford learned graph-free attention).

Table S1: Instantiations across domains. The spatial mixer is the modular slot; choice depends on regime (long T + sparse adjacency favors fixed-spectral; dense local mesh admits attention).

Domain (model)	Local features $\rightarrow$ coefficients	Spatial mixer (slot)	Temporal solver	Params
River (SepGFNO)	Reach attrs $\rightarrow$ routing coeffs	Upstream sum $\mathbf{NQ}$ (fixed)	FFT-IIR	27K
PDE (SepFNO)	Boundary feats $\rightarrow$ mixing weights	Chebyshev $T_k(\mathbf{A})\mathbf{h}$ (fixed)	FFT	454K
PDE (SepTransolver)	Boundary feats $\rightarrow$ slice tokens	Physics-Attention (learned, graph-free)	windowed I/O	630K
Traffic (SepGRU)	Sensor stats $\rightarrow$ hop weights	$\sum \alpha_k \mathbf{A}^k \mathbf{h}$ (fixed)	GRU	51K
Citation (SepHybrid)	Text feats $\rightarrow$ hidden state	Pre-smoothing $\mathbf{A}^K \mathbf{x}$ (fixed)	–	–

B Applicability Checklist

Table S2 summarizes which of the six domains we evaluate satisfy the two necessary conditions of §3.5 (graph-independent features; graph is connectivity only) and whether the separation principle works empirically. The four positive domains (river, PDE, traffic, citation) satisfy both criteria; the two negative controls (power flow, airport networks) violate one each.

Table S2: Applicability checklist across the six domains evaluated.

Criterion	River	PDE	Traffic	Citation	Power	Airport
Graph-independent features?	Yes	Yes	Yes	Yes	Partial	No
Graph is connectivity only?	Yes	Yes	Yes	Yes	No	Yes
Separation works?	Yes	Yes	Yes	Yes	No	No

C Additional River Results

Figure S1 shows that high reach-wise R^2 extends across both basins and from headwaters to mainstem reaches. The intermediate drainage-area bins have a somewhat broader error distribution, so the strong basin-wide median does not imply uniform accuracy at every reach.

The common-cohort metrics in Table 2 intersect the outputs of all three models at 86,133 active reaches. After $t_0 = 200$, each reach has 2,400 hourly steps; only finite NSE values and target coefficient of variation above 0.05 are retained. Peak error uses the target window maximum and searches the predicted peak within ± 24 h. The anchor-only control bypasses learned coefficient corrections but retains the five-regime architecture with untrained components; it is not a pure

deterministic Muskingum–Cunge solver. Its 1 h timing median matches the trained model, while magnitude and shape differ substantially. Persistence omits known time-varying lateral forcing and is not the primary comparator for this emulation task. Operational trunk injection nodes receive reference flows in both SepGFNO and GraphSAGE+GRU evaluations; these results therefore measure boundary-conditioned transfer, not fully closed-loop upstream propagation. The separate trained-only diagnostic covers 86,152 active reaches, with NSE P5 0.961, 97.1% above 0.9, and 98.7% positive.

Table S3: Mississippi common-cohort relative L_2: median [P10, P90] across 86,133 reaches. High flow restricts each reach to times at or above its target 98th percentile.

Model	All times	High flow
SepGFNO, trained	0.0280 [0.0120, 0.0809]	0.0240 [0.0088, 0.0686]
SepGFNO, anchor-only	0.5011 [0.4998, 0.5194]	0.5045 [0.5000, 0.5477]
GraphSAGE+GRU, ep150	0.3606 [0.0700, 1.5143]	0.3410 [0.0415, 0.8570]

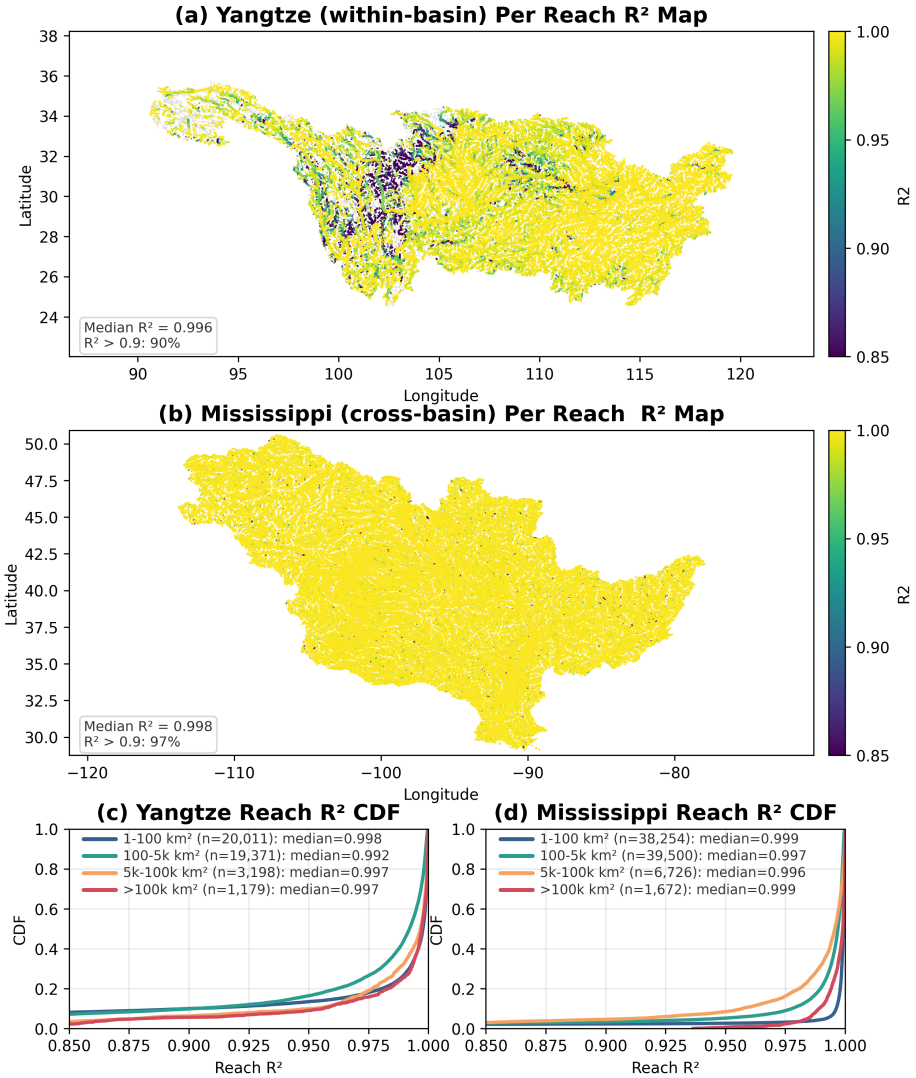

Figure S1: Reach-wise performance using Yangtze-trained SepGFNO weights. Per-reach R^2 maps for (a) Yangtze and (b) Mississippi, and R^2 by upstream drainage area for (c) Yangtze and (d) Mississippi. High accuracy spans small and large drainage areas, with somewhat wider spread in intermediate tributaries.

D Boundary Error Localization (PDE)

Figure S2 shows where cross-geometry residual error concentrates. We partition the test mesh (triangle+notch, 1187 nodes) into three zones: *boundary* (nodes on the mesh boundary, $n = 133$), *near-boundary* (within $\sim$2 mesh layers of the boundary, $n = 636$), and *interior* (the rest, $n = 418$). Per-node relative L_2 is averaged over 30 evaluation samples on the 4-geometry checkpoints of both SepFNO (matched-aug) and SepTransolver (no aug, the per-architecture best).

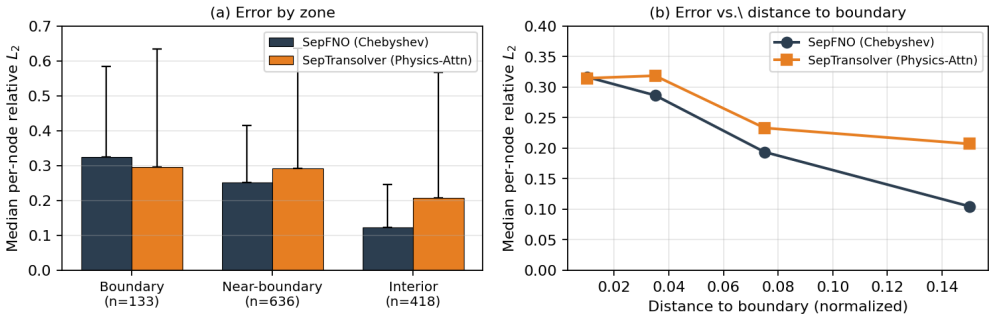

Figure S2: Cross-geometry error patterns differ structurally between the two spatial mixers. (a) Median per-node relative L_2 by zone (whisker = 75th percentile). SepFNO's Chebyshev polynomial mixing concentrates error at the boundary (boundary/interior ratio $\sim$2.7$\times$); SepTransolver's Physics-Attention slice tokens distribute error far more uniformly (ratio $\sim$1.4$\times$). (b) Median error vs. normalized distance to boundary: SepFNO drops $\sim$3$\times$ from boundary to deep interior (0.32$\rightarrow$0.10), while SepTransolver drops only $\sim$1.5$\times$ (0.31$\rightarrow$0.21). Both architectures struggle most near boundaries, but the magnitude of locality bias differs: fixed local-polynomial mixing is locality-sensitive in its errors; global-attention mixing spreads error uniformly across the mesh.

E Architecture Details

E.1 SepGFNO: River Routing

Physical basis: Muskingum–Cunge routing. The Muskingum–Cunge (MC) method [Bindas et al., 2024] is a finite-difference solution to the one-dimensional continuity equation for open-channel flow, routing discharge through a river network reach by reach. For a single reach, the MC equation relates outflow Q at time t to inflow I from upstream and lateral inflow q' from the local catchment:

$$Q_i(t) = c_{1,i}\, I_i(t) + c_{2,i}\, I_i(t{-}1) + c_{3,i}\, Q_i(t{-}1) + c_{4,i}\, q_i'(t) \tag{2}$$

where the coefficients c_1–c_4 encode the reach's hydraulic properties. These coefficients are derived from Manning's equation and channel geometry. Following the differentiable MC framework (δMC) [Bindas et al., 2024, Ji et al., 2025], channel flow velocity is computed from Manning's equation:

$$v = \frac{1}{n}\, d^{2/3}\, S_0^{1/2}, \quad d = \left(\frac{Q\, n\, (q_s{+}1)}{p_s\, S_0^{1/2}} \right)^{\frac{3}{5+3q_s}} \tag{3}$$

where n is Manning's roughness, S_0 is bed slope, d is flow depth, and p_s, q_s are hydraulic geometry parameters relating top width $w = p_s\, d^{q_s}$ to depth [Bindas et al., 2024]. Wave celerity $c = \frac{5}{3}v$ (v clamped to $[0.3, 15]$ m/s) determines the wave travel time $K_w = \Delta x / c$, from which the MC coefficients follow:

$$c_1 = \frac{\Delta t - 2K_w X}{D}, \quad c_2 = \frac{\Delta t + 2K_w X}{D}, \quad c_3 = \frac{2K_w(1{-}X) - \Delta t}{D}, \quad c_4 = \frac{\Delta t}{D} \tag{4}$$

where $D = 2K_w(1{-}X) + \Delta t$, $X = 0.3$ is the storage coefficient, and $\Delta t = 3600$ s. Mass conservation requires $c_1 + c_2 + c_3 = 1$.

In δMC, the coefficients are computed sequentially—each reach routes flow from its immediate upstream neighbor, proceeding from headwaters to outlet in topological order. This sequential formulation, while physically accurate, cannot exploit GPU parallelism and requires $O(D \cdot T)$ sequential operations for a network of diameter D.

Matrix reformulation for GPU parallelism. SepGFNO reformulates the reach-by-reach routing as simultaneous matrix operations over the entire network. The upstream connectivity is encoded in a binary matrix $\mathbf{N} \in \{0,1\}^{m \times m}$ where $N_{ij} = 1$ if reach j is an immediate upstream neighbor of reach i (with appropriate normalization for confluence nodes). The inflow to each reach is then:

$$\mathbf{I}(t) = \mathbf{N}\,\mathbf{Q}(t) \tag{5}$$

where $\mathbf{Q}(t) \in \mathbb{R}^m$ is the vector of all reach discharges at time t. Since $\mathbf{N}$ is strictly lower-triangular when nodes are sorted by upstream area (headwaters first), the matrix power $\mathbf{N}^D = \mathbf{0}$ for D equal to the graph diameter. Starting from an initial guess (e.g., headwater-only routing), iterating Eq. (2) with $I_i(t) = (\mathbf{N}\mathbf{Q}(t))_i$ converges to the exact sequential solution in D iterations. Each iteration is a single sparse matrix–vector multiply, fully parallelizable on GPU.

Step 1 (Encoder). An MLP maps local reach attributes $\mathbf{a}_i = (S_0, \Delta x, n, q_s, p_s, \text{uparea}, \ldots) \in \mathbb{R}^{N_{\text{feat}}}$ to a hidden vector $\mathbf{h}_i = \text{MLP}_{\text{enc}}(\mathbf{a}_i) \in \mathbb{R}^d$, shared across all reaches and basins.

Step 2 (Coefficient prediction with physics anchors). For each of K flow regimes (corresponding to quantiles of the flow distribution: low, medium, high), physics-anchored MC coefficients $\tilde{c}_{1,k}, \ldots, \tilde{c}_{4,k}$ are computed from Manning's equation (Eq. (3)–(4)) using representative discharge values. These anchors are then modulated by learned correction heads:

$$c_{2,k,i} = \sigma(\mathbf{W}_2\mathbf{h}_i)_k \cdot 2 \cdot \tilde{c}_{2,k,i}, \quad c_{1,k,i} = 1 - c_{2,k,i} - c_{3,k,i} \quad \text{(mass conservation)} \tag{6}$$

When the MLP output is zero, $\sigma(0) \cdot 2 = 1$ and the coefficients match the physics exactly. The MLP learns corrections to account for sub-grid heterogeneity and model structural error. All MLP weights are graph-independent—the same parameterization scheme applies to any river basin.

Step 3 (FFT-IIR temporal processing). The MC equation (Eq. (2)) is a first-order IIR (infinite impulse response) filter with feedback coefficient c_3. Its impulse response is $h[t] = c_3^t$ for $t \geq 0$. Given an input signal $u_i(t) = c_1 I_i(t) + c_2 I_i(t-1) + c_4 q_i'(t)$, the output is a convolution:

$$Q_{k,i}(t) = (u_i * h_k)[t] = \sum_{\tau=0}^{t} u_i(\tau)\, c_{3,k,i}^{t-\tau} \tag{7}$$

This convolution is computed in the frequency domain:

$$H_k = \text{FFT}(c_3^0, c_3^1, \ldots, c_3^{T-1}), \quad \mathbf{Q}_k = \text{IFFT}(\text{FFT}(\mathbf{u}_k) \odot H_k) \tag{8}$$

processing all T timesteps in $O(T \log T)$ with **no backpropagation through time**. This is the key efficiency gain: standard autoregressive GRU/LSTM baselines require $O(T)$ sequential steps with full BPTT memory.

When an initial condition $Q_i(0) = Q_{\text{init}}$ is available (warm start), the homogeneous solution $Q_{\text{homo}}(t) = c_3^{t+1} \cdot Q_{\text{init}}$ is added to the convolution output analytically.

Step 4 (Iterative spatial propagation). Starting from headwater-only solutions (Eq. (8) with $I = 0$), we iterate:

$$\mathbf{I}^{(\ell)}(t) = \mathbf{N}\,\mathbf{Q}^{(\ell-1)}(t) \tag{9}$$

$$\mathbf{Q}^{(\ell)} = \text{FFT-IIR}\big(c_1\mathbf{I}^{(\ell)} + c_2\mathbf{I}_{t-1}^{(\ell)} + c_4\mathbf{q}',\ c_3\big) + \mathbf{Q}_{\text{homo}} \tag{10}$$

for $\ell = 1, \ldots, D$ (graph diameter). All K regimes are batched: $\mathbf{Q}^{(\ell)} \in \mathbb{R}^{K \times m \times T}$ is reshaped to $\mathbb{R}^{m \times KT}$ for a single sparse matrix multiply per iteration, then reshaped back. Gradient checkpointing is used during training to bound memory to $O(m \cdot T)$ regardless of D.

Step 5 (Regime blending). A dilated temporal CNN processes the mean routed discharge (detached from the computation graph) to predict time-varying regime weights:

$$w_k(t) = \text{softmax}_k\left(\text{CNN}\left(\frac{\bar{Q}(t)}{\bar{\bar{Q}}}\right)\right), \quad Q_{\text{final}}(t) = \sum_{k=1}^{K} w_k(t)\, Q_k(t) \tag{11}$$

The CNN uses dilations $[1, 2, 4, 8]$ with kernel size 7, capturing temporal context at multiple scales. Optional temporal smoothing (average pooling with kernel size 25) reduces high-frequency oscillations in regime weights. The K-regime system allows the model to represent nonlinear stage-dependent routing behavior—low flows route slowly (high c_3, strong attenuation), while high flows route fast (low c_3, near-kinematic wave)—without explicit flow-dependent coefficient update, which would break the FFT-IIR approach.

Subgraph training, leaf transfer, and cascade inference. Large river networks can have graph diameters far beyond the number of spatial propagation iterations that is practical during training. We therefore distinguish three evaluation modes. A *leaf subgraph* is a complete upstream drainage subtree terminating at a selected outlet reach; it has no missing upstream boundary and can be routed independently. Leaf-transfer tests train on leaf subgraphs from one basin and evaluate on held-out leaf subgraphs from another basin, measuring whether the learned local routing coefficients generalize across hydroclimatic and geomorphic settings. For full-basin inference, the network is decomposed into bounded-diameter leaves and trunk segments. A *trunk segment* is a mainstem piece whose upstream boundary receives inflow from upstream leaves or earlier trunk segments. In the reported *operational cascade*, segments are processed from upstream to downstream, with reference discharge supplied at trunk injection nodes. This avoids running spatial propagation over the full basin diameter while providing known boundary conditions; the reported metrics do not measure fully closed-loop propagation of upstream prediction error. All experiments use the same learned SepGFNO weights; only the fixed connectivity matrix $\mathbf{N}$, local reach attributes, and boundary inflows change across basins.

Summary. The model has 27,924 parameters (K=5 regimes, hidden dim 64). All learned weights (encoder MLP, coefficient heads, temporal CNN) are graph-independent. The only graph-dependent quantity is $\mathbf{N}$, which is a binary connectivity matrix carrying no learned parameters. To deploy on a new basin, one simply constructs $\mathbf{N}$ from the new river network topology and provides the local reach attributes—no retraining, fine-tuning, or adaptation required.

E.2 SepFNO: PDE on Irregular Meshes (Navier–Stokes)

Step 1 (Lift). Per-node boundary features $\mathbf{b}_i \in \mathbb{R}^6$ (is_boundary, distance, normal, curvature, density—*no absolute coordinates*) and input vorticity $\omega_i(t)$ are lifted: $\mathbf{h}_i(t) = \mathbf{W}_{\text{lift}}[\mathbf{b}_i; \omega_i(t)]$.

Step 2 (FNO layers, $\times L$). Each layer applies four operations:

(a) Temporal FFT: $\hat{\mathbf{h}}_i(f) = \text{FFT}_t(\mathbf{h}_i(t))$ for frequency modes $f = 0, \ldots, \lfloor T/2 \rfloor$.

(b) Spectral kernel (graph-free): $\hat{\mathbf{h}}'(f) = \mathbf{W}^{(\ell)}_{\text{spec}}[\text{Re}(\hat{\mathbf{h}}(f)); \text{Im}(\hat{\mathbf{h}}(f))]$, where $\mathbf{W}_{\text{spec}}$ is shared across all nodes and graphs.

(c) Chebyshev spatial mixing (fixed graph operator):

$$T_0(\mathbf{A}) = \mathbf{I}, \;\; T_1(\mathbf{A}) = \mathbf{A}, \;\; T_k(\mathbf{A}) = 2\mathbf{A}T_{k-1}(\mathbf{A}) - T_{k-2}(\mathbf{A}) \tag{12}$$

$$\mathbf{s}_i(f) = \mathbf{W}_{\text{spat}}[T_0\hat{\mathbf{h}}'_i(f); T_1\hat{\mathbf{h}}'_i(f); \ldots; T_K\hat{\mathbf{h}}'_i(f)] \tag{13}$$

where $\mathbf{W}_{\text{spat}} \in \mathbb{R}^{d\times(K+1)d}$ is graph-independent. The Chebyshev polynomials use the normalized adjacency (spectrum in $[-1, 1]$), eliminating per-graph λ_{max} rescaling.

(d) Inverse FFT + residual: $\mathbf{h}_i^{(\ell+1)}(t) = \text{GELU}(\text{IFFT}_f(\hat{\mathbf{h}}'_i + \mathbf{s}_i) + \mathbf{W}^{(\ell)}_{\text{res}}\mathbf{h}_i^{(\ell)}(t))$.

Step 3 (Projection). $\hat{\omega}_i(t) = \text{MLP}_{\text{proj}}(\mathbf{h}_i^{(L)}(t))$.

Training schedule and regularization. We train SepFNO with AdamW (initial LR 10^{-3}, weight decay 10^{-4}) for 300 epochs under a cosine LR schedule that decays to zero, followed by a fine-tuning phase of up to 100 additional epochs at a constant LR of 10^{-5}. The fine-tuning phase consistently improves cross-geometry L_2 by an additional $\sim$5% relative to the cosine endpoint, because by the end of the cosine schedule the LR is too small to make further progress while the loss is still slowly decreasing; switching to a constant low LR releases this residual signal without re-entering the noisier high-LR regime. Four regularization mechanisms are used jointly during training, each addressing a distinct leak channel identified in Section 3: (i) *rotation augmentation* applies random $\theta \in [0, 2\pi)$

rotations and reflections to node positions, surface normals, and the vorticity sign, breaking absolute orientation cues; (ii) *boundary-feature dropout* randomly zeros entire boundary feature columns with probability $p = 0.3$, preventing the model from memorizing specific boundary-shape signatures; (iii) ω*-scale augmentation* multiplies input vorticity by a per-sample lognormal factor $\exp(\pm 0.2)$, breaking the implicit correlation between vorticity magnitude and geometry identity; (iv) ω*-noise* adds Gaussian noise of std 0.05 to the input field, providing local smoothing. These augmentations are designed to be compatible with the fixed-Chebyshev spatial operator (which is magnitude-agnostic by construction); applied to Transolver under matched settings, the same augmentations cause divergence (Appendix G).

E.3 SepGRU: Traffic Forecasting (Learned Dynamics)

Task and protocol. We follow the DCRNN traffic forecasting protocol [Li et al., 2018]. Each model observes 12 historical 5-min speed values and predicts the next 12 steps. The cross-city transfer setting trains all learnable parameters on METR-LA and evaluates directly on PEMS-BAY. The reported main metric is the 30-min MAE on PEMS-BAY. Oracle gap is computed relative to a model trained and evaluated on PEMS-BAY under the same forecasting protocol.

Input features and zero-shot convention. At each sensor i and time t, the dynamic input is

$$\mathbf{x}_i^t = [\mathrm{speed}_i^t, \mathrm{tod}_i^t], \tag{14}$$

where tod is the time-of-day encoding. SepGRU-v2 also uses local sensor statistics $\mathbf{f}_i \in \mathbb{R}^5$ to predict diffusion weights. These statistics are computed from the available unlabeled time series of the deployment city and contain no target training labels. Thus, the traffic experiment is a zero-label, cross-city transfer protocol: the target graph and simple target sensor summaries are available, but no PEMS-BAY supervised training is performed.

Step 1 (Shared temporal encoder). Each sensor history is encoded independently by a shared GRU:

$$\mathbf{h}_i^{(\mathrm{enc})} = \mathrm{GRU}_{\mathrm{enc}}(\mathbf{x}_i^{1:T}). \tag{15}$$

The same GRU weights are applied to every sensor in both cities. No sensor identity or graph-specific parameter enters this encoder.

Step 2 (Local-feature diffusion coefficients). SepGRU-v2 predicts node-wise multi-hop diffusion weights from local sensor statistics:

$$\boldsymbol{\alpha}_i = \mathrm{softmax}(\mathrm{MLP}_{\mathrm{coeff}}(\mathbf{f}_i)) \in \mathbb{R}^{K+1}. \tag{16}$$

Here $\alpha_{i,0}$ weights the sensor's own hidden state and $\alpha_{i,k}$ weights the k-hop diffused hidden state. The coefficient MLP is shared across sensors and cities, so the node-wise adaptation is feature-conditioned rather than identity-conditioned.

Step 3 (Decoder with fixed graph diffusion). At each prediction step τ, a shared GRUCell updates the temporal hidden state:

$$\mathbf{h}^{(\tau)} = \mathrm{GRUCell}(\hat{\mathbf{y}}^{(\tau-1)}, \mathbf{h}^{(\tau-1)}). \tag{17}$$

Spatial information is then mixed using fixed powers of the normalized road adjacency $\mathbf{A}$:

$$\mathbf{h}_i^{(\mathrm{out})} = \alpha_{i,0}\mathbf{h}_i^{(\tau)} + \sum_{k=1}^{K} \alpha_{i,k}(\mathbf{A}^k\mathbf{h}^{(\tau)})_i, \qquad \hat{y}_i^{(\tau)} = \mathbf{W}_{\mathrm{proj}}\mathbf{h}_i^{(\mathrm{out})}. \tag{18}$$

The adjacency matrix changes from METR-LA to PEMS-BAY, but it is never learned; it only appears through fixed matrix multiplications. All trainable weights—the GRU, coefficient MLP, and projection layer—are shared across graphs.

Baseline variants. SepGRU-v1 removes the learned coefficient predictor and uses fixed hop weights, testing whether node-wise adaptation is needed. NodeEmbedGRU adds a learnable embedding for each training sensor, testing the effect of source-node-specific parameters. EntangledGRU places graph diffusion inside the recurrent cell while keeping parameters shared, testing whether graph use inside temporal dynamics alone breaks transfer. The main result is that NodeEmbedGRU fails most severely, while EntangledGRU remains close to SepGRU, indicating that source-specific parameters are more damaging than fixed graph diffusion itself.

F GraphSAGE+GRU Stabilization Attempts

Earlier dense GraphSAGE+GRU configurations showed instability or out-of-memory failures, including a 5,000-node inference probe. A later sparse-COO, node-budgeted implementation with OOM retry trained through epoch 150 and supports full-basin evaluation. The dense probe does not bound the stabilized implementation's inference capability. As a shared-weight inductive model, its parameters remain meaningful on another basin.

Table S4: Controlled one-epoch Yangtze workload on Tesla V100S-PCIE-32GB: 1,388 subgraphs, 70,726 reach instances, and 1,200 hourly steps. Peak allocated GPU memory is reported. These are same-work timings, not matched-accuracy convergence times.

Model	Seconds/epoch	GPU GB	Parameters
SepGFNO (FFT-IIR)	72	2.5	27,924
SepGFNO (sequential IIR)	3,121	5.4	27,924
GraphSAGE+GRU (sparse, ep150)	396	24.2	67,651

The sequential control retains the same coefficient heads, five regimes, sparse upstream propagation, propagation rounds, and blending CNN as FFT-IIR; only the temporal recurrence is evaluated with a loop. Both SepGFNO solvers report one-epoch loss 0.1395. The historical sequential JSON parameter count of zero is a wrapper artifact, so the shared architecture count is shown. The sparse GraphSAGE+GRU comparison tests implementation-level efficiency under this workload; its cross-basin accuracy is in Table 2.

G SepTransolver: Adaptation and Multi-Geometry Stability

Canonical Transolver [Wu et al., 2024] is a single-mesh, single-time-index method (forward pass: spatial coordinates + function values + scalar time index $\rightarrow$ field at one target time). It has no multi-geometry training mode. We construct **SepTransolver** by slotting Transolver's Physics-Attention spatial mixer into our space-time factorization protocol via five changes:

1. **Time-windowed channel I/O**: scalar time index replaced by T_w-channel input/output (in_dim $= n_{\text{nf}} + T_w$, out_dim $= T_w$); predicts an entire window in one forward pass, making the model time-parallel (non-recurrent).
2. **Graph-free boundary features** replace absolute spatial coordinates (x, y), closing the Level-2 leak (§3).
3. **Multi-mesh shared-weight batching** across heterogeneous meshes within a single batch (canonical Transolver processes one mesh per forward pass).
4. **Removal of scalar-t conditioning** that would leak training-distribution time signature.
5. **Stability-tuned learning rate**: empirically required to be $\sim$30$\times$ smaller than canonical defaults to avoid divergence at multi-geometry training.

We do not claim Physics-Attention is new; our contribution is a factorization protocol that adapts Physics-Attention for zero-shot cross-geometry transfer from single-geometry training.

Learning-rate sensitivity. At canonical-style LR (1×10^{-4}), SepTransolver diverges to NaN within 30–90 epochs at $\geq$ 9 training geometries (Table S5, top rows). We swept LR $\in \{1\times10^{-4}, 3\times10^{-5}, 1\times10^{-5}\}$, gradient clipping $\in \{1.0, 5.0\}$, and augmentation on/off. LR$= 3\times10^{-5}$ trains stably across all multi-geometry configurations. SepFNO is stable at LR$= 1\times10^{-3}$ ($\sim$30$\times$ larger), reflecting the activation-scale sensitivity of softmax attention vs. polynomial mixing.

Architecture-tuned augmentation. Our SepFNO leak-control augmentation suite (rotation, boundary-feature dropout, ω-scale, ω-noise) is *harmful* for SepTransolver in the original notch experiment. At 4g (LR$= 3\times10^{-5}$), matched-aug yields L_2=0.052, no aug yields L_2=0.046 (-12%). The ω-scale/ω-noise components are the primary culprits; intermediate (no-ω) configuration yields L_2=0.049. We attribute this to the activation-scale sensitivity of Physics-Attention's softmax slice tokens. This historical notch sweep is distinct from the later five-target, matched-augmentation sweep in Figure 3.

Table S5: Earlier triangle+notch SepTransolver configuration sweep, distinct from the final five-target scaling table. *Default LR* (1×10^{-4}): diverges at $\geq$ 9g. *Tuned LR* (3×10^{-5}): stable across these configurations.

Config	N	LR	Aug	Final L_2	Status
SepTransolver, default LR	4	1e-4	matched	0.045	stable
SepTransolver, default LR	9	1e-4	matched	—	NaN ep 50 (best 0.038)
SepTransolver, default LR	15	1e-4	matched	—	NaN ep 50 (best 0.056)
SepTransolver, tuned LR	4	3e-5	matched	0.052	stable
SepTransolver, tuned LR	4	3e-5	none	**0.046**	stable
SepTransolver, tuned LR	9	3e-5	matched	0.034	stable
SepTransolver, tuned LR	9	3e-5	none	**0.022**	stable
SepTransolver, tuned LR	15 (bigger)	3e-5	matched	0.031	stable
SepTransolver, tuned LR	15 (bigger)	3e-5	none	**0.021**	stable
SepFNO base	9	1e-3	matched	0.031	stable
SepFNO base	15	1e-3	matched	0.030	stable
SepFNO bigger	15	1e-3	matched	0.028	stable

Mechanism we hypothesize but do not prove. Physics-Attention uses a softmax over slice tokens weighted by a learned temperature. Multi-geometry training varies the magnitude scale across batches; the softmax is sensitive to this, leading either to peaked attention (vanishing gradients on off-peak slices) or large activations (overflow). Smaller LR shrinks per-step updates enough to avoid the unstable regime, and removing ω-scale aug eliminates an extra source of magnitude perturbation. SepFNO's spatial operator is a fixed polynomial of the adjacency—no softmax, no learned temperature—so magnitude variation scales its outputs linearly without instability.

Earlier diverged runs (with mislabelled aug presets). An earlier set of Transolver runs used the `-sink` flag, which we initially documented as a sink-token attention stabilizer. Code inspection (`transolver_baseline.py`) revealed that `-sink` is an augmentation-preset that overwrites flags to a stronger setting (ω-scale 0.3, ω-noise 0.1). The runs in Table S5 use the corrected flag set; earlier runs are superseded.

H Related Cross-Geometry PDE Benchmarks and How Our Setup Differs

Operator learning on irregular geometries is an active area, and several benchmarks evaluate generalization across mesh, geometry, or parameter shifts. Here we survey the most relevant efforts and indicate where each differs from the multi-geometry knowledge-accumulation regime evaluated in §4.2.

Position relative to existing work. The broad area of geometry generalization in PDE learning is well established. What we are not aware of, and what we therefore construct in §4.2, is a controlled setup that specifically tests *cross-geometry knowledge accumulation*: train jointly on many distinct mesh topologies, evaluate zero-shot on a held-out geometry, and measure how performance scales with the number and diversity of training geometries. This is the scaling-with-N-training-geometries axis (Fig. 3), and it is the axis where our principle's value becomes empirically visible.

On the fairness of baseline adaptations. We do not claim that the architectures evaluated in Table 3 are intrinsically incapable of cross-geometry transfer. Their published validation regimes — single physical systems with mesh adaptation, OOD shifts within an engineering family, obstacle-shape variation within cylinder flow — differ from ours, and our results reflect performance under the specific adaptations we made (e.g., adding boundary-feature inputs, channel-style time windows, multi-mesh batching). For Transolver specifically, the adaptation is structural (the original takes a scalar time index and predicts a single field; we extend this to time-windowed channel I/O), and we report the adaptation in detail under the name *SepTransolver*. Reviewers and readers who can identify suboptimalities in our adaptations will materially improve the comparison; every adapted launch script and the model code are available at `https://github.com/chaopengshen/gsno_crossgraph`.

Table S6: Related cross-geometry / mesh-generalization benchmarks. None tests how cross-geometry error scales with the number and diversity of training geometries, the regime our multi-target scaling experiment targets.

Benchmark / paper	Scope	How it differs from our regime
PDEBench [Takamoto et al., 2022]	Broad SciML benchmark for time-dependent PDEs (Burgers, Navier–Stokes, etc.) with standardized data, baselines, metrics	Tests model performance on each PDE family separately; not designed for joint training across many distinct geometries with zero-shot transfer to a held-out geometry.
SURF [Künzli et al., 2024]	GNN fluid simulator generalization across mesh resolution, parameter ranges, topology shifts within engineering systems (valves, turbomachinery)	Within-system OOD generalization rather than cross-distinct-shape knowledge accumulation; does not run the "train on N shapes, ask if more N helps" protocol that motivates our scaling figure.
Schmöcker et al. [2024]	MGN trained on cylinder-flow with various obstacle shapes and multi-object configs	Cross-shape transfer within a single physical setup (cylinder flow); the shape variation is over obstacle inserts, not fundamentally different mesh topologies (T-beam, hourglass, star, etc.) as in our setup.
Geo-FNO [Li et al., 2023a]	FFT operators on irregular geometries via learned deformations to latent regular grids	A method/architecture, not a standardized benchmark; the deformation is learned per-shape and time is advanced autoregressively.
GINO [Li et al., 2023b]	Geometry-aware neural operator for large-scale 3D PDEs	A method paper with bespoke geometry-aware encoders; spatial operators carry mesh-dependent learned weights (different from our graph-free constraint).
Transolver [Wu et al., 2024]	Physics-Attention slice mechanism for irregular meshes	A method paper; published evaluations are within a single mesh distribution per task. We adapt to our cross-geometry windowed protocol as *SepTransolver* (§G).

I Citation Network Experiments

We test the separation principle on citation network node classification using the ArnetMiner dataset (GDABench [Liu et al., 2024]): DBLP, ACM, and Citation graphs with 6,775-dimensional bag-of-words text features and multi-class labels (5 research areas).

SepHybrid architecture. SIGN-style [Frasca et al., 2020] pre-smoothing ($\mathbf{x}_{\text{aug}} = [\mathbf{x}, \mathbf{A}\mathbf{x}, \mathbf{A}^2\mathbf{x}, \ldots, \mathbf{A}^K\mathbf{x}]$) provides multi-scale graph features. An MLP processes the concatenated input, followed by 3 iterative refinement layers: $\mathbf{h} \leftarrow \mathbf{h} + \text{MLP}_k([\mathbf{h}; \mathbf{A}\mathbf{h}])$. All learned parameters are graph-free; the graph enters only through fixed matrix powers during pre-smoothing and iterative steps.

Table S7: Citation cross-graph classification (Macro-F1 %). All 6 source→target pairs.

Model	D→A	D→C	A→D	A→C	C→D	C→A	Avg
SepHybrid	**59.1**	**65.8**	**61.7**	**68.9**	67.3	66.0	**64.8**
GCN-3L	57.0	65.3	58.5	66.9	**67.9**	**66.7**	63.7
GCN-2L	56.3	64.0	59.8	67.0	66.9	66.0	63.3
MLP	41.6	45.6	47.0	47.2	50.2	45.2	46.1

SepHybrid (64.8%) outperforms GCN-3L (63.7%) by +1.1 points, winning 4 of 6 transfer pairs. The result is modest compared to the spatiotemporal domains, consistent with the observation that static node classification with homophilic graphs provides less room for improvement through separation.

J NSE Geometries and Data

Table S8 varies the number of *training* NSE geometries: 5/6/7/9/12 for SepFNO and 6/7/9/15 for SepTransolver. Its held-out evaluation set is the same five newly generated car, cup, spiral, pistol, and saw geometries for every row; none appears in a training pool, and there is no target fine-tuning. Their meshes have 1,803, 981, 3,817, 786, and 1,930 nodes. The handoff used viscosity 0.01, quasi-periodic forcing amplitude 5, 50 samples, $t_{\text{end}} = 9$, 1,000 solver steps, and every-fourth-step saving ($T = 251$). The final SepFNO results use seeds 101, 202, and 303; its 15g single run and earlier single-run curve are excluded. SepTransolver is a separate single-run companion under matched augmentation. The table merges per-target means with the final pool summary; the SepFNO uncertainty column is seed SD, not across-target SD. Exact pool memberships, final checkpoint filenames, and launch configurations were not supplied with the reported table.

Table S8: Varying-size NSE training pools, evaluated zero-shot on the same five held-out car/cup/spiral/pistol/saw geometries (relative vorticity L_2). SepFNO target cells are three-seed means; SepTransolver is single-run.

Model	Pool	Car	Cup	Spiral	Pistol	Saw	Mean $\pm$ seed SD
SepFNO	5g	.0652	.1109	.1668	.0563	.0623	$.0923 \pm .0037$
	6g	.0717	.1147	.1759	.0614	.0675	$.0982 \pm .0065$
	7g	.0591	.1073	.1544	.0511	.0549	$.0854 \pm .0068$
	9g	.0426	.0997	.1177	.0394	.0360	$.0671 \pm .0014$
	12g	.0431	.0969	.1176	.0379	.0359	$.0663 \pm .0005$
SepTransolver	6g	.0490	.1352	.1938	.0817	.0468	.1013
	7g	.0495	.1298	.1754	.0763	.0473	.0957
	9g	.0316	.1201	.1195	.0489	.0268	.0694
	15g	.0285	.1087	.1174	.0467	.0248	.0652

At fixed SepTransolver pool size, adding sinusoidal channel to a 4g base changes five-target mean error from 0.0994 to 0.0754, adding star to 0.0827, and adding hourglass to 0.1102. Thus composition may help or hurt. Target-only oracle errors reported with matched handoff settings are 0.0175 (car), 0.0314 (cup), 0.0326 (spiral), 0.0135 (pistol), and 0.0171 (saw); best reported cross-geometry/oracle ratios span 1.45–3.60. These oracle comparisons do not establish a universal near-parity result.

Table S9 tests a different protocol: fixed checkpoints trained on the same four base geometries (ellipse with triangle hole, mesh rectangle with triangle hole, rectangle with triangle and circular holes, and six-petal flower with circular hole) are evaluated without adaptation on all six other geometries in a separate ten-geometry pool. These six held-out meshes are not the five car/cup/spiral/pistol/saw targets of Table S8. Vector velocity is reconstructed for predictions and targets from $\nabla^2\psi = -\omega$ with zero boundary streamfunction, $u = \partial_y\psi$, and $v = -\partial_x\psi$. Relative velocity L_2 combines nodes, output times, and both components per window, then averages over windows. Pressure is eliminated and neither predicted nor stored.

Table S9: Fixed four-geometry NSE training set; zero-shot evaluation on its six held-out geometries from the ten-geometry pool. Each cell is relative vorticity L_2 / reconstructed velocity L_2. This test set differs from Table S8.

Target	SepFNO	MLP_FNO	GCN_FNO	GraphSAGE_FNO	MeshGraphNet
L-shape	.0478/.0258	.1356/.0768	.2964/.1859	.0813/.0439	.2479/.1488
Crescent	.0324/.0192	.0885/.0576	.2201/.1774	.0441/.0237	.1894/.1557
Hourglass	.0539/.0291	.1547/.0882	.2887/.1794	.0971/.0535	.2433/.1387
Sinusoidal	.0563/.0400	.0901/.0684	.2669/.2245	.0581/.0412	.2283/.1929
Star	.0586/.0323	.1113/.0740	.2653/.2005	.0643/.0394	.2329/.1725
Notch	.0403/.0237	.1026/.0700	.2759/.1962	.0593/.0376	.2397/.1681
Mean	.0482/.0283	.1138/.0725	.2689/.1940	.0674/.0398	.2303/.1628

On the external Schmöcker unsteady cylinder-flow benchmark [Schmöcker et al., 2024], models train on `standard_cylinder` and evaluate on that geometry and zero-shot on `cylinder_stretch`. The matched intersection has 32 simulations (11 with Reynolds number below 47); SepFNO's broader

evaluation had 401 tasks while the MeshGraphNet evaluation covered tasks 0–39. Errors are raw v_x RMSE over 20 steps. The native MeshGraphNet checkpoint was at epoch 20 and undertrained (one-step error 11.7×10^{-3} versus the published 2.32×10^{-3}); therefore this comparison does not establish superiority over a converged baseline. High-Re SepFNO errors show phase drift in the supplied diagnosis, rather than numerical NaNs.

Table S10: External benchmark on 32 matched simulations: v_x RMSE $\times 10^{-3}$ as median / mean. Standard is in distribution; stretch is zero-shot geometry transfer.

Re bin	n	MGN standard	SepFNO standard	MGN stretch	SepFNO stretch
< 47	11	121.1 / 114.7	53.2 / 52.3	163.0 / 163.3	76.0 / 143.4
47–100	8	136.0 / 111.1	142.5 / 195.3	147.4 / 165.1	73.7 / 235.1
≥ 100	13	145.1 / 124.5	128.8 / 222.1	160.0 / 178.9	376.4 / 303.6
All	32	136.5 / 117.8	53.6 / 157.0	159.5 / 170.1	125.7 / 231.4

For the 4-geometry training setting used in Table 3, the training meshes are *Ellipse with Triangle Hole*, *Mesh Rectangle with Triangle Hole*, *Rectangle with Triangle and Circular Holes*, and *Six-Petal Flower with Circular Hole*. The held-out zero-shot test mesh is *Triangular with a Notch*.

Figure S3 shows representative vorticity rollouts on five of the training/test geometries; the full gallery of all 15 training meshes plus the held-out test mesh is shown in Figure S4. We use 2D incompressible Navier–Stokes on irregular Delaunay-triangulated domains ($\sim$1,200–1,700 nodes per mesh) with interior obstacles. A quasi-periodic body force (amp $= 5$, $\nu = 0.01$) drives complex spatiotemporal dynamics on each mesh, generating multi-step vorticity rollouts; the cross-geometry experiments train on N training geometries and hold out triangle+notch for zero-shot evaluation.

K Code Availability

Code is available at `https://github.com/chaopengshen/gsno_crossgraph`.

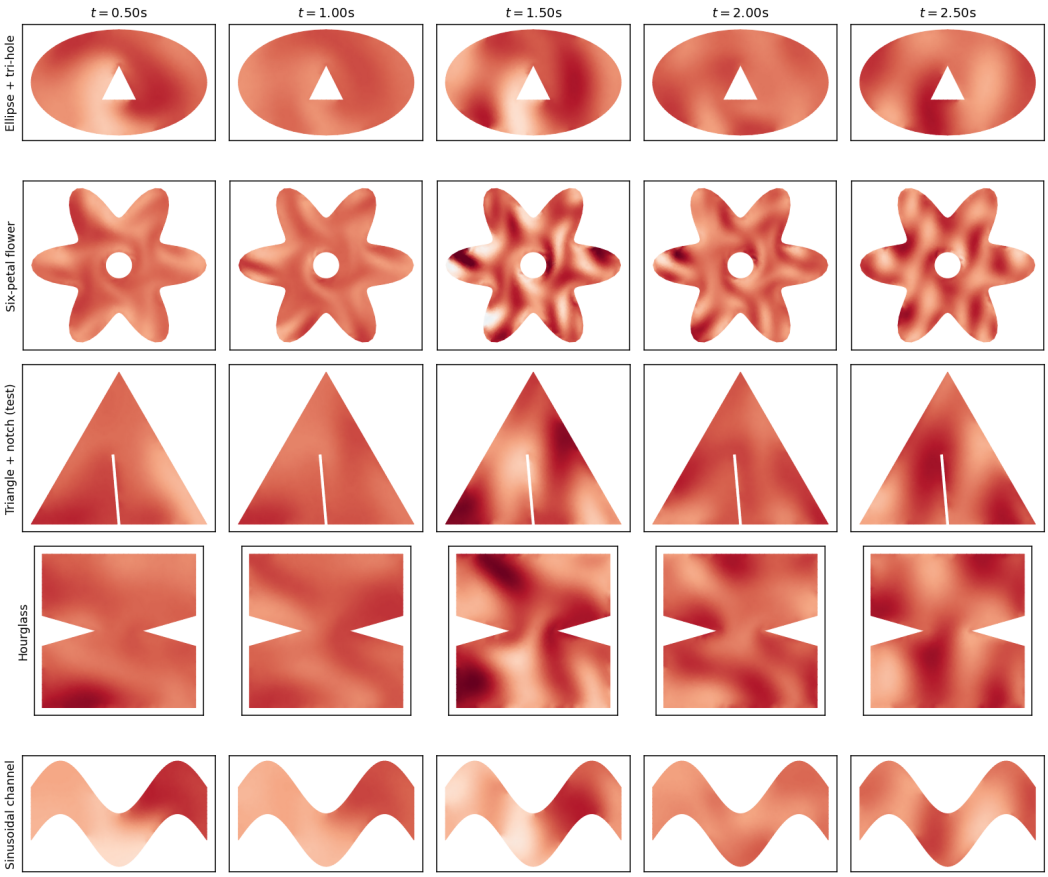

Figure S3: NSE vorticity fields on five training/test geometries (ellipse+triangle hole, six-petal flower, triangle+notch test, hourglass, sinusoidal channel) at five time instants. Quasi-periodic forcing drives complex spatiotemporal dynamics on each irregular mesh. The triangle+notch (third row) is the held-out zero-shot test geometry.

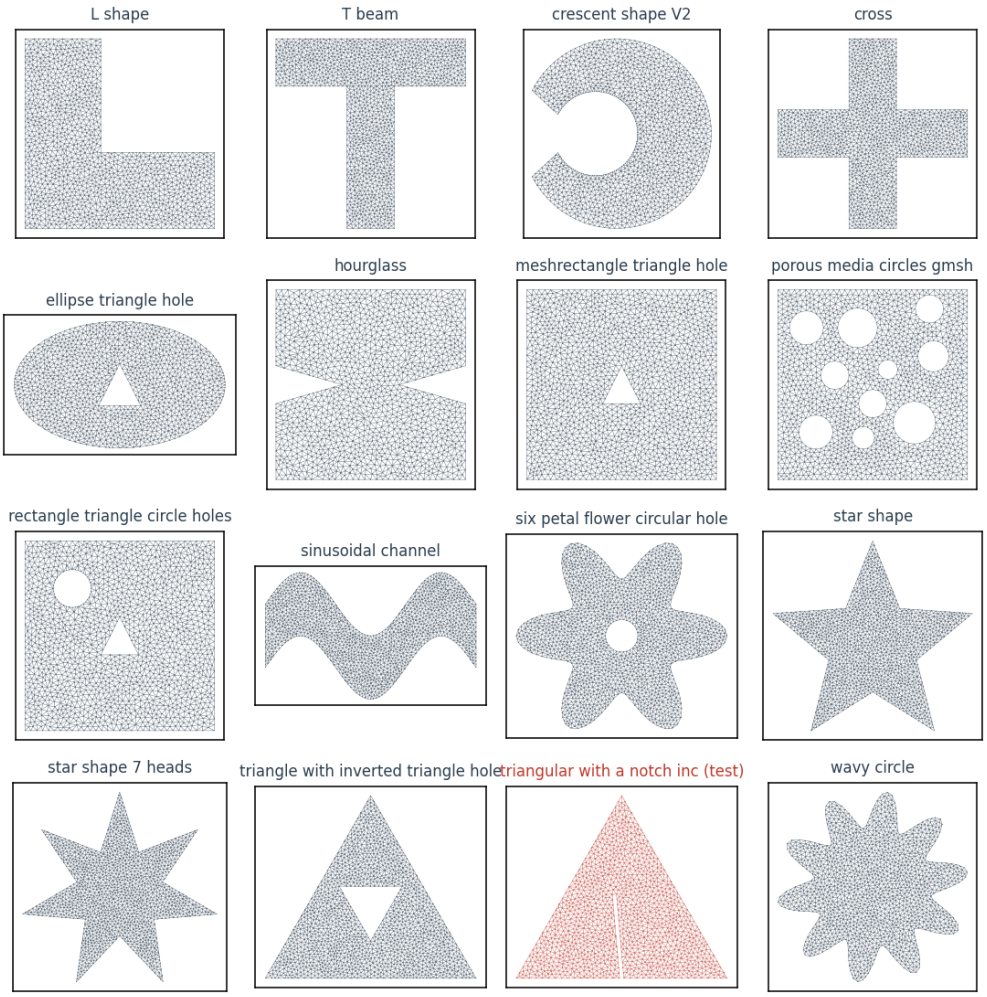

Figure S4: All 16 NSE meshes used in the cross-geometry experiments: 15 training geometries (black mesh) and the held-out test geometry triangle+notch (red, bottom row). Mesh size varies $\sim$1,200–1,700 nodes; geometries include simple convex shapes (ellipse, hexagon), shapes with internal obstacles (multi-holes, porous media), and shapes with sharp boundary features (hourglass, star, crescent, sinusoidal channel).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contribution as space-time factorization for graph dynamics, with claims supported by experiments on river routing, irregular-geometry Navier–Stokes, traffic forecasting, citation classification, and two negative controls. The scope is also qualified through an applicability criterion and negative-control domains.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations in the conclusion, including the remaining PDE cross-geometry gap, the limits of linear temporal kernels for state-dependent spatial coupling, and the need for additional regimes or hybrid physics-informed extensions. The negative controls further clarify cases where the proposed factorization is not expected to apply.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper proposes a design principle and architectural factorization, but does not present formal theorems requiring proof. The mathematical expressions are used to define the model structure and domain instantiations rather than to state theoretical guarantees.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The manuscript provides substantial architectural and experimental details, including model components, datasets, graph sizes, major hyperparameters, and evaluation protocols.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Data are public. Code is available at `https://github.com/chaopengshen/gsno_crossgraph`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the main training and evaluation settings for each domain, including river subgraph/full-basin evaluation, the traffic protocol, PDE geometry splits and augmentation settings, and citation cross-graph transfer. Additional architecture and training details are provided in the appendices.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

   Justification: Experiments are deterministic once a trained checkpoint and graph split are fixed.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the main compute setting for several experiments, including V100 32GB hardware, per-epoch time, peak GPU memory, and scalability limits for major baselines.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses scientific, traffic, citation, and benchmark graph datasets and does not involve human-subject experiments, private personal data, or high-risk generative models. The work follows the NeurIPS Code of Ethics and preserves anonymity in the submission.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses positive impacts through faster and more transferable modeling for river routing, traffic forecasting, and physical simulation. It also discusses risks from over-reliance on zero-shot predictions in high-stakes settings, especially when deployment graphs violate the stated applicability criteria.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk generative models, scraped personal datasets, or models with obvious dual-use misuse risks requiring controlled release safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper cites the datasets and baseline methods from other literature.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The submission does not introduce a new public dataset, benchmark, or released model asset beyond the paper itself.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing, user studies, or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human-subject research, crowdsourcing, or collection of participant data.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were not used as an important, original, or non-standard component of the core method, experiments, data generation, or evaluation.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.