PRIORS OVER SYSTEMS: SYNTHETIC PRETRAINING FOR SPATIO-TEMPORAL FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Spatio-temporal foundation models aim to transfer across networks, but current pretraining relies on a limited collection of real systems. We ask whether a sufficiently expressive generative prior can replace real-system data collection with synthetic pretraining. MICROFLOW generates synthetic systems in three separated stages: it draws a latent constitution that fixes how the system may behave, simulates its evolution from micro-level interactions, and then adds an observer that aggregates the granular trajectories into time series. We find that this separation of dynamics from observation makes the prior more expressive than naive functional series propagation. To demonstrate the usefulness of MICROFLOW, we develop MICROFLOW-ST, an architecture that models each modality (time series and graphs) with its own tower and couples them for spatiotemporal training. Without observing any real system during pretraining, our frozen checkpoints are competitive with similarly-sized state-of-the-art foundation models trained on both synthetic and real-world data, outperforming all of the ones we evaluate on zero-shot long-term spatiotemporal forecasting. Lightweight adaptation of MICROFLOW-ST further lowers its MAE by 33% and 42% on average across short- and long-horizon tasks, respectively. We release the synthetic generator and all reported checkpoints.

1 INTRODUCTION

While spatio-temporal forecasting has traditionally meant fitting a separate model per structure (or network), spatio-temporal foundation models (STFMs) instead aim to transfer a single pretrained model across systems. In this work, we consider complex systems defined jointly by their structure and by their behavior, as observed through time series. Classical specialist models couple temporal operators to a supplied or learned structure, but are optimized for a single network (Li et al., 2018; Wu et al., 2019). STFMs such as UNIST, OPENCITY, and FACTOST-V2 extend this interface across structures and tasks (Yuan et al., 2024; Li et al., 2024; Zhong et al., 2026). However, what needs to be transferred is an entire system—a network, its interaction rules, and the instruments that measure it—and public pretraining corpora contain far fewer distinct systems than temporal windows. Repeated windows improve estimation on networks already seen but do not introduce new underlying generation mechanisms, interactions, or measurement schemes. To make progress, we must therefore collect data that is more representative of the diversity of real-world systems.

If distinct systems are the scarce resource, one option is to specify them rather than collect them. A generative prior over systems becomes a sampler, not a fixed collection, so coverage is bounded by what the prior can express rather than by what has been instrumented and released. Prior-Data Fitted Networks (PFNs) make this substitution operational: trained on purely synthetic datasets drawn from such a prior, a PFN learns to predict the hidden part of a task directly from its observed part, in a single forward pass, without fitting any parameters to that task (Müller et al., 2022). This principle has produced effective priors for individual modalities, e.g. TabPFN for tabular data (Hollmann et al., 2025), TempoPFN for time-series (Moroshan et al., 2025), and GraphPFN for graphs (Eremeev et al., 2026). Although their inputs are structured, each work only introduces a unimodal prior: one principal observable interface and a prior tailored to its completion task.

A spatio-temporal prior is more demanding than any one of these interfaces. The model observes heterogeneous temporal and structural views, an association between measurements and entities,

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

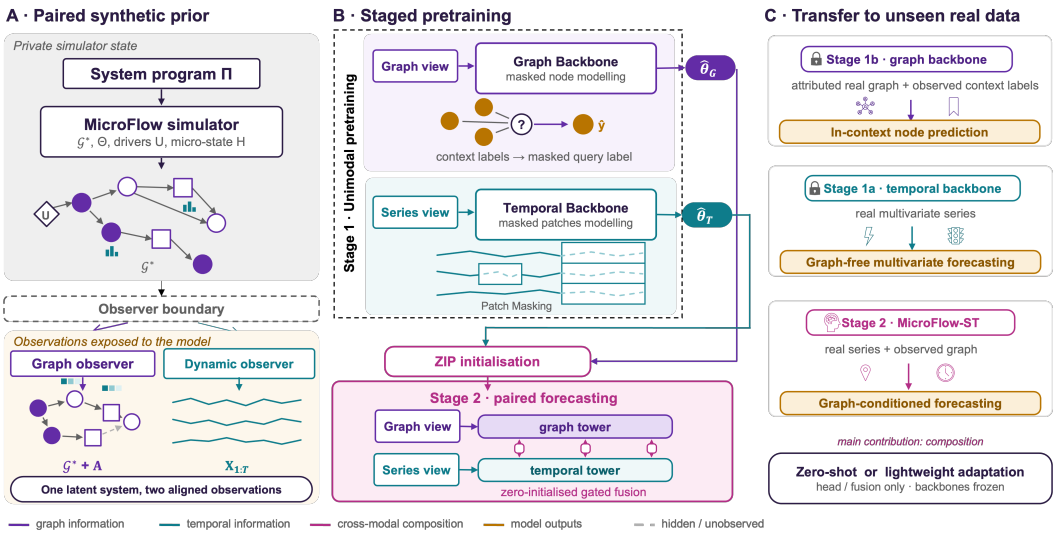


Figure 1: Overview of MICROFLOW and MICROFLOW-ST. **A:** MICROFLOW samples a private system program and simulator state, then exposes aligned graph and temporal observations to the model. **B:** graph and temporal towers are pretrained independently in Stages 1b and 1a, then zipped and composed through zero-initialized gated fusion for paired forecasting in Stage 2. **C:** the resulting checkpoints transfer to unseen real data through in-context node prediction, graph-free multivariate forecasting, and graph-conditioned forecasting, either zero-shot or with lightweight adaptation.

and a forecast query whose answer may depend on their coupling. Independent time-series and graph draws could teach useful marginals but no meaningful and generalizable cross-modal correspondence. Conversely, generating dynamics directly on observed graph channels identifies entities, state variables, and instruments by construction. A useful spatio-temporal prior must therefore generate not only both marginals, but also their alignment through a shared latent system: which entities interact, how their state evolves, and how that evolution becomes observable.

We ask whether such a prior can be built, and whether a model trained only on its samples transfers to real systems. For the former, we introduced a prior called MICROFLOW that models time-series as observations of aggregated micro-interactions: it samples a system constitution and its admissible rules, simulates marked micro-interactions under exogenous forcing, and only then aggregates the resulting trajectories into observed time series. To learn over the generated data, we introduce MICROFLOW-ST, a synchronized dual-tower STFM in which the temporal and structural backbones exchange information via cross-modal, graph-conditioned bridges after every block. Its staged recipe first learns the unimodal inference paths, joins them with initially non-perturbing fusion, and then learns their paired coupling. Figure 1 summarizes the architecture and deployment pipeline.

One frozen MICROFLOW-ST checkpoint trained only on MICROFLOW samples transfers to real traffic networks, performing competitively against STFMs pretrained on both synthetic and real data. We then use controlled experiments to confirm the importance of different components of our approach, including the mechanistic generation, the coupling in the architecture, and the staged training procedure. Beyond this, our contributions are:

- **A mechanistic prior over observed systems.** MICROFLOW formalizes a constitution–interaction–observation separation in which channels are observations of a shared micro-level trajectory rather than variables in the interaction graph.
- **A compositional spatio-temporal architecture and recipe.** MICROFLOW-ST combines factorized temporal and structural towers with repeated sparse bridges, no-op gated composition, and staged marginal-to-joint training.
- **Controlled synthetic-to-real evidence.** Frozen transfer, a matched-prior comparison, restricted adaptation, and staged-versus-scratch training isolate the value and scope of the prior, coupling mechanism, and training recipe on six real traffic systems.

2 Related Work

Spatio-temporal forecasting. Graph forecasting models combine temporal operators with propagation over supplied or learned relations, including recurrent diffusion, graph convolutions, adaptive adjacency, and node-conditioned parameters (Li et al., 2018; Yu et al., 2018; Wu et al., 2019; Bai et al., 2020). Recent foundation models broaden transfer across cities: UniST uses prompts over urban modalities (Yuan et al., 2024), OpenCity jointly pretrains across traffic systems (Li et al., 2024), and FactoST-v2 separates a transferable temporal model from target-specific spatial adaptation (Zhong et al., 2026). These approaches establish the value of cross-system pretraining on real urban corpora. Our setting asks whether a reusable temporal model and a rule for consuming unseen structure can instead be learned without observing a real pretraining system.

Time-series foundation models on system data. Large pretrained forecasters make frozen prediction a practical alternative to per-dataset training. The Toto family uses decoder-only multivariate Transformers (Cohen et al., 2025; Khwaja et al., 2026); TimesFM uses a patched decoder trained on a large real corpus (Das et al., 2024); Moirai combines a multi-domain corpus with masked any-variate modeling (Woo et al., 2024); and Chronos casts augmented series as token sequences (Ansari et al., 2024). We retain these models as graph-agnostic references on the same traffic tasks: they measure the transferable value of temporal and multivariate context, while MicroFlow-ST additionally learns how to compose that context with an observed network.

Synthetic structured data. PFNs learn an inference algorithm from tasks sampled from a procedural prior (Müller et al., 2022). More precisely, leveraging in-context learning, a new dataset is supplied as a **context** at inference time, and predictions for **query** points are obtained in a single forward pass by conditioning on this context. This principle has transferred to real tabular prediction with TabPFN (Hollmann et al., 2025), forecasting with ForecastPFN and TimePFN (Dooley et al., 2023; Taga et al., 2025), and node prediction with GraphPFN (Eremeev et al., 2026); simulation has likewise supported transfer in physical systems (Arndt et al., 2025). NeoST samples stochastic dynamics on graph nodes and fits a closed-form latent adapter within each inference window (Feng et al., 2026). MicroFlow instead separates a typed system constitution, marked micro-interactions, and aggregate measurement. The resulting checkpoint is used frozen, and can optionally be finetuned.

3 Problem Setup

We model a dataset as a partial observation of a hidden system $\mathcal{S}$. The model receives time series $\mathbf{X} \in \mathbb{R}^{K \times T}$, where K is the number of observed measurement channels and T the number of time steps, together with, when available, an attributed graph $\mathcal{G} = (V, E, \mathbf{A}^V)$. Here V is the set of entities constituting the system, $E \subseteq V \times V$ a set of heterogeneous relations between them, and $\mathbf{A}^V \in \mathbb{R}^{|V| \times F}$ the matrix of F entity attributes. A relation $\mathcal{R} \subseteq \{1, \dots, K\} \times V$ associates measurements with entities, so that $(k, v) \in \mathcal{R}$ means channel k observes entity v. Crucially, K need not equal $|V|$: one entity may carry several measurements, and one measurement may aggregate several entities. The graph $\mathcal{G}$ is side information about the observed system, while the measurements $\mathbf{X}$ are partial observations of its behaviour.

Following the PFN framework, we express both forecasting and graph prediction as masked completion. Every entry of $\mathbf{X}$ and of $\mathbf{A}^V$ is either unmasked (**context**), and therefore visible to the model, or masked (**query**) and withheld from it; binary masks $\mathbf{M}^X \in \{0,1\}^{K \times T}$ and $\mathbf{M}^A \in \{0,1\}^{|V| \times F}$ record which. Writing $\mathcal{C}$ for the unmasked entries together with the structure $(V, E, \mathcal{R})$, and $\mathcal{Q}$ for the masked ones, a model with parameters θ predicts

$$p_\theta(\mathcal{Q} \mid \mathcal{C}) . \tag{1}$$

Different tasks are different masks over the same interface: masking a suffix of $\mathbf{X}$ gives forecasting, masking a column of $\mathbf{A}^V$ on some entities gives node prediction from the remaining entities' labels, and masking whole rows of $\mathbf{X}$ gives imputation of unobserved channels.

4 MICROFLOW: A PRIOR OVER OBSERVED SYSTEMS

4.1 FROM CHANNEL PRIORS TO SYSTEM PRIORS

A naïve way to generate multivariate time series is to define each observed channel directly as a function of the observed channels at the previous time step. Let $x_{k,t}$ be the value of channel k at time t, and let $\mathbf{x}_{\cdot,t-1} = (x_{1,t-1}, \ldots, x_{K,t-1})$ be the vector of all channels at time $t-1$. For each channel k, this approach specifies a propagation function f_k and a noise term $\epsilon_{k,t}$, and sets

$$x_{k,t} = f_k(\mathbf{x}_{\cdot,t-1},\ \epsilon_{k,t})\,. \tag{2}$$

In other words, the observed channels drive each other directly, and no latent variables are involved. Effective priors depart from this form by routing dependence through latent variables instead of through the observed channels. Taga et al. (2025) give expressive priors over multivariate series by drawing latent functions from Gaussian processes and mixing them, so channels correlate because they share hidden causes. Moroshan et al. (2025) sample from a mixture of generator families and apply a cascade of nonlinear transforms in the univariate setting.

Neither construction, however, produces everything a prior needs in order to cover real spatio-temporal systems. We identify three properties such a prior must have:

1. **Cyclic causality.** A quantity must be able to influence its own future. In real spatio-temporal systems, effects combine as they spread through the structure and return to where they originated, so dependence is recurrent rather than hierarchical. Priors built on structural causal models rule this out, because their dependencies are acyclic by construction.
2. **Multivariate observability.** One entity must be able to carry more than one measurement, since a sensor typically reports several quantities at once. A prior that generates one series per node can only do this by splitting the entity into several nodes, which makes two readings of the same thing look like two things interacting.
3. **Latent topology.** The graph must carry information about the system that the series alone do not give away. In functional priors the graph is instead the recipe used to compute the series and is discarded afterwards, so it can be read off the data it produced and the model learns nothing from being handed it.

MICROFLOW obtains all three by simulating the micro-level interactions a system is made of, and then measuring the result with a separate observation layer. Section 4.2 shows where each property comes from.

4.2 A THREE-LEVEL CONSTRUCTION

Constitution. We first sample a system constitution

$$\mathcal{S} = (\Pi, \mathcal{G}^*, \Theta) \tag{3}$$

Π holds the high-level choices for this system: which topology families to combine and which kind of dynamics to run on them. $\mathcal{G}^*$ is the topology those choices produce, a directed graph over the system's entities along which units move. It shapes the dynamics without reproducing them: units travel along its edges, but what a channel ends up recording depends on the queues, delays, and capacities they meet on the way and on how the measurement is taken. This is what gives the prior a latent topology (the third target property above) since the graph constrains the observations without being recoverable from them. Θ holds the numerical parameters attached to that topology, governing how fast units move, where they accumulate, and how they are measured. The constitution therefore fixes which interactions are possible and how they are constrained, without treating the observed channels as the state that evolves.

Micro-level interaction. A system is also driven from outside. We sample *exogenous drivers* $\mathbf{U}_{1:T}$: series that affect the system but are not affected by it, such as demand arriving at its boundary. These are drawn from a library of stochastic, periodic, and regime-switching components that produces highly diverse patterns (Appendix B.3).

A particle simulator then runs the constitution under that forcing:

$$\mathbf{H}_{1:T} = \mathsf{Sim}(\mathcal{S}, \mathbf{U}_{1:T}). \tag{4}$$

$\mathbf{H}_t$ is the full internal state of the simulator at time t: where every unit is and how its attributes have evolved. Because this state is carried forward step by step, the same quantity can influence its own future, which gives the prior the cyclic causality (the first target property above) without imposing an acyclic ordering on the observed channels.

Observation. The simulator's state is never shown to the model. Instead, sampled observers read it and write out the time series $\mathbf{X}_{1:T}$, explained in the problem settings (Section 3). The simplest observer counts the units passing through one entity per time step; others average or take quantiles of unit attributes, and others still read groups of entities or combine several measurements, so one channel may summarize a whole region of the system (Appendix B.4). Because observers are sampled independently of the entities they read, one entity can carry several channels and one channel can aggregate several entities, while all channels remain functions of the same underlying trajectory. This gives the prior multivariate observability (the second target property above): several measurements of one entity are several views of one state, not separate entities that happen to interact.

The remaining inputs of Section 3 are produced alongside $\mathbf{X}$. The graph $\mathcal{G}$ shown to the model is a perturbed view of $\mathcal{G}^*$, with some true edges removed and some spurious ones added, so that the supplied structure is informative but not exact. The attribute matrix $\mathbf{A}^V$ is built from the entity parameters in Θ, distorted and re-encoded to yield attribute matrices of arbitrary width and mixed type, both categorical and continuous. Finally, the masks $\mathbf{M}^X$ and $\mathbf{M}^A$ are drawn to define the task, splitting every entry into the context $\mathcal{C}$ and the query $\mathcal{Q}$ of Section 3: random patches of each series are hidden, at its horizon and in its interior, while entity attributes are split into visible context slots and withheld query slots. The exact perturbation, distortion, and masking distributions are given in Appendix B.

5 MICROFLOW-ST

Our generator produces two paired modalities for every system: a graph and a panel of multivariate time series, aligned because they observe the same simulation. MICROFLOW-ST is factorized along the same split. We build a temporal and a structural backbone separately, each trained on its own modality's tasks, then merge them with learned cross-modal layers. Keeping the modalities in separate backbones means each one can be evaluated, and reused, on its own: the temporal backbone is a forecaster that needs no graph, and the structural backbone is a graph model in its own right (Appendix D). It also means a further modality would be added by training another backbone and another set of bridges, rather than by redesigning the model.

Figure 1B shows how Stage 2 copies both pretrained backbones into a single *synchronized dual tower*: the two towers have the same depth, so at every layer each one advances in its own representation space and a sparse bridge then exchanges information between them along the observed graph. Repeating the exchange at every layer rather than once at the end allows information exchange between modalities after every block.

Schema-independent tokenization. The temporal stream follows the TOTO family (Cohen et al., 2025; Khwaja et al., 2026), designed to accommodate the flexible, highly variable distributions found in observability data. Each series is scaled using its unmasked history, transformed with asinh, split into patches, and embedded together with the masks that say which patches are withheld and which samples are missing. The graph stream represents each entity as a short row of tokens: its attributes, plus learned tags encoding each attribute's type, plus a randomized orthogonal identifier (Kim et al., 2022). The identifier gives each entity a distinct vector carrying no meaning of its own, so attention can tell that two tokens concern the same entity without any fixed semantics attached to a position in the graph. This is what lets the model use multivariate observability: several channels reading one entity are resolved as views of that entity, and the association relation $\mathcal{R}$ can be consumed for any schema. Exact embeddings, limits, and dimensions appear in Appendix A.

Two factorized towers. Each modality's tokens form a grid: for the temporal stream, series by patches; for the structural stream, entities by attributes. Attending over such a grid as one flat

sequence costs the square of its total size. Both towers instead factorize that attention, alternating along one axis and then the other. In the temporal tower, attention runs first within a row, one series across its patches, and then down a column, one patch position across series, which gives a multivariate route that does not need a graph. In the structural tower a row is one entity across its attributes and a column is one attribute across all entities, so the column direction learns how an attribute is distributed over the population. This second pattern is deliberately topology-free, as a tabular in-context model would be.

Writing $\mathbf{Z}^\ell \in \mathbb{R}^{K\times J\times d}$ for the temporal states at depth ℓ, with $J = \lceil T/P \rceil$ patches of size P and embedding width d, and $\mathbf{W}^\ell \in \mathbb{R}^{|V|\times F\times d}$ for the structural states, the two towers advance independently at every depth:

$$\widetilde{\mathbf{Z}}^\ell = \mathsf{T}_\ell(\mathbf{Z}^\ell), \qquad \widetilde{\mathbf{W}}^\ell = \mathsf{G}_\ell(\mathbf{W}^\ell), \tag{5}$$

where T_ℓ and G_ℓ each apply the two-axis pattern above within their own modality. Neither reads the graph: up to this point the two streams have not exchanged anything.

Sparse, gated fusion. The two streams are then merged by a bridge F_ℓ, which produces the states entering the next depth:

$$(\mathbf{Z}^{\ell+1}, \mathbf{W}^{\ell+1}) = \mathsf{F}_\ell(\widetilde{\mathbf{Z}}^\ell, \widetilde{\mathbf{W}}^\ell; E, \mathcal{R}). \tag{6}$$

Cross-modal attention is factorized in the same spirit as within the towers. Rather than letting every temporal token attend to every structural token, F_ℓ decides what each token may read from the two structural inputs of Section 3: a series reads the entities it observes under $\mathcal{R}$ and their neighbors under E, while an entity reads its neighbors and the series attached to them. Each token therefore attends to a small, system-specific set rather than to everything, and connectivity enters the model only here. Appendix A gives how these sets are capped and packed for computational efficiency.

Task heads. Rather than predicting a single value, the temporal head emits several quantiles of the predictive distribution at once, trained with the pinball loss, which gives a probabilistic forecast instead of a point estimate. The same technique predicts masked continuous entity attributes but using independent quantile heads; masked categorical attributes use a shared classification head trained with cross-entropy. One further incidence head is used only when pretraining the structural backbone. Recall from Section 4 that the graph shown to the model is a perturbed view of the true topology, containing both real and spurious edges: this incidence head scores entity pairs and is trained to tell the two apart, so the backbone learns to judge which supplied connections are consistent with the rest of the system. Details about the output heads are described in Appendix A

5.1 Training strategy

We train in two stages so that temporal forecasting, graph completion, and their composition do not compete from initialization; Figure 1B shows the corresponding initialization and parameter flow. Let $\theta_T, \theta_G, \theta_F$ denote the temporal backbone, graph backbone, and graph-routed bridge parameters respectively. All stages use only MicroFlow data, i.e. $(\mathcal{D}^T_{\text{syn}}, \mathcal{D}^G_{\text{syn}}, \mathcal{D}^{ST}_{\text{syn}}) \sim p_{\text{M}}$ with p_{M} the MicroFlow prior.

$$\widehat{\theta}_T = \text{Train}_{1a}(\theta_T; \mathcal{D}^T_{\text{syn}}), \tag{7}$$

$$(\widehat{\theta}_G, \widehat{\theta}^G_F) = \text{Train}_{1b}(\theta_G, \theta^{G\leftrightarrow G}_F; \mathcal{D}^G_{\text{syn}}), \tag{8}$$

$$\widehat{\theta}_{ST} = \text{Train}_2\Big(\text{Zip}_0(\widehat{\theta}_T, \widehat{\theta}_G, \widehat{\theta}^G_F, \theta^{G\leftrightarrow T}_F); \mathcal{D}^{ST}_{\text{syn}}\Big). \tag{9}$$

Stage 1a trains the temporal stream on suffix forecasting and interior reconstruction, with random univariate and multivariate attention masks. **Stage 1b** trains the graph stream to reconstruct masked continuous and categorical node attributes, infer sparsely revealed task-specific labels, and distinguish observed from faulty incidence. These checkpoints define independently deployable temporal and graph priors.

Stage 2 copies both towers into the full model, zip-initializes new cross-modal branches as gated no-ops, and jointly optimizes all parameters for forecasting. It draws paired tasks from the same temporal family as Stage 1a, removes privileged exogenous drivers and their edges, and does not continue the graph auxiliary losses. The reported checkpoint retains metric nodes and fixed scalar gates.

Table 1: Zero-shot traffic forecasting. Each cell is MAE/RMSE in raw data units (lower is better). Avg. is the unweighted mean over the six displayed datasets. Bold and underlined values are best and second-best within each model family and task. *NEOST fits a closed-form ridge adapter to every input window. †Suspected contamination from overlap between the pretraining corpus and evaluation data; full details are provided in the appendix.

Family	Model (stage)	Params	Pretraining	METR-LA	PEMS-BAY	PEMS03	PEMS04	PEMS07	PEMS08	Avg.
Long-term forecasting ($96 \rightarrow 96$)										
STFM	**MICROFLOW-ST (2)**	39.3M	Synthetic	**7.934**/13.886	5.544/10.712	**84.089**/121.300	**104.100**/143.708	**124.889**/**170.159**	**90.551**/126.379	**69.52**/97.69
STFM	NEOST *	89.26M	Synthetic	8.31/**13.70**	**4.86**/**8.14**	84.49/**107.85**	105.25/**137.35**	144.18/175.65	94.03/**121.87**	73.52/**94.09**
STFM	FACTOST-V2	4.3M	Real + synth.	10.87/21.49	6.11/11.56	86.90/124.62	107.15/147.60	126.83/173.31	91.09/127.28	71.49/100.98
STFM	FACTOST-V1	1.3M	Real	12.68/18.82	6.12/10.47	113.62/144.80	142.11/175.99	156.07/190.59	121.43/151.59	92.01/115.38
STFM	OPENCITY	1.67M	Real	10.85†/17.60†	6.92†/11.79†	125.18/159.23	153.18†/188.95†	172.20/211.25	128.48†/161.75†	99.47/125.10
STFM	UNIST	2.42M	Real	25.88/30.19	9.29/12.72	102.87/137.90	115.76/153.58	144.67/184.62	104.77/136.21	83.87/109.20
TSFM	**MICROFLOW-ST (1a)**	15.8M	Synthetic	**7.747**/**13.703**	5.387/10.453	87.597/125.439	105.266/145.925	127.322/173.242	91.449/**127.511**	70.79/**99.38**
TSFM	TOTO 2.0	313M	Real + synth.	10.237/18.911	5.217/10.463	81.561/125.356	**100.926**/149.410	129.829/191.351	**90.358**/134.966	**69.69**/105.08
TSFM	CHRONOS-2†	120M	Real + synth.	8.521/15.660	**5.015**/**10.118**	93.270/136.242	110.430/154.855	129.119/179.799	96.528/138.436	73.81/105.85
TSFM	MOIRAI †	91.4M	Real	12.87/22.19	6.51/11.70	**74.85**/**110.78**	105.95/**141.40**	**125.91**/**169.43**	119.10/151.83	74.20/101.22
Short-term forecasting ($12 \rightarrow 12$)										
STFM	**MICROFLOW-ST (2)**	39.3M	Synthetic	4.483/8.827	2.659/6.284	**24.938**/40.221	32.650/51.572	37.742/58.239	26.805/42.590	21.55/34.62
STFM	NEOST *	89.26M	Synthetic	4.90/**8.21**	2.67/4.80	25.16/**34.90**	**30.98**/**43.63**	**35.70**/**49.48**	**26.28**/**36.42**	**20.95**/**29.57**
STFM	FACTOST-V2	4.3M	Real + synth.	5.32/12.82	2.27/5.46	25.70/40.75	33.13/49.92	37.85/57.48	27.11/41.18	21.90/34.60
STFM	FACTOST-V1	1.3M	Real	5.98/14.26	**2.02**/**4.59**	30.12/46.92	38.65/56.59	45.47/66.68	32.14/47.27	25.73/39.39
STFM	OPENCITY	1.67M	Real	**4.30**†/8.37†	3.23†/6.91†	30.37/47.49	39.34†/58.50†	45.18/67.15	32.45†/48.42†	25.81/39.47
STFM	UNIST	2.42M	Real	24.33/29.31	14.89/16.92	101.87/129.94	67.91/87.63	104.93/133.65	73.46/93.14	64.57/81.77
TSFM	**MICROFLOW-ST (1a)**	15.8M	Synthetic	**4.176**/**8.328**	2.288/5.418	**22.171**/**35.404**	**28.891**/**44.902**	**32.932**/50.968	**23.312**/**36.954**	**18.96**/**30.33**
TSFM	TOTO 2.0	313M	Real + synth.	6.566/12.223	2.451/5.395	23.072/37.704	31.198/48.822	34.877/54.950	24.772/39.380	20.49/33.08
TSFM	CHRONOS-2†	120M	Real + synth.	4.630/9.872	2.342/5.548	25.524/41.397	32.465/49.680	37.148/57.730	26.703/41.109	21.47/34.22
TSFM	MOIRAI †	91.4M	Real	5.55/13.79	**1.97**/**4.69**	28.23/46.36	34.65/52.13	35.64/**50.25**	38.23/53.12	24.05/36.72

Full objectives, masking procedures, dimensions, optimization settings, and adaptation scopes are given in Appendix A; corpus and task construction are described in Appendix B.

6 EXPERIMENTS

6.1 EXPERIMENTAL SETUP

We evaluate on six established five-minute traffic networks used by recent STFM studies (Zhong et al., 2026; Feng et al., 2026). METR-LA and PEMS-BAY measure speed at 207 and 325 sensors (Li et al., 2018); PEMS03, PEMS04, PEMS07, and PEMS08 measure flow at 358, 307, 883, and 170 sensors (Guo et al., 2019; Song et al., 2020). Sensors are graph nodes and processed road relations supply edges. We use the standard chronological 70/10/20 split for METR-LA and PEMS-BAY and 60/20/20 for the four PEMS flow datasets, with complete split-local windows following BasicTS recipes (Shao et al., 2024).

We report short ($12 \rightarrow 12$) and long ($96 \rightarrow 96$) forecasting. Each example contains the complete sensor panel and road graph; MAE and RMSE are pooled over finite targets in raw units. For adaptation, we use the final 10% of complete training origins for eight epochs, select by full-validation loss, and apply no dataset-specific hyperparameter search. Appendix C gives origin counts, masks, optimization scopes, and complete protocol details. We measure the effect of each component by running ablations within our own implementation: matched synthetic-prior comparison, model inference modes, model adaptation regimes, and staged versus scratch training.

Beyond spatio-temporal forecasting, Appendix D evaluates the graph stream (Stage 1b) on seven GraphLand node-property tasks spanning regression, binary, and multiclass targets, both frozen and under restricted adaptation, against graph foundation models pretrained on real graphs. These tasks are unrelated to traffic, so they test whether the structural marginal learned from the synthetic prior generalizes beyond the transport systems. Among graph foundation models, the frozen checkpoint is competitive without having observed a real graph during pretraining.

6.2 FROZEN ZERO-SHOT TRANSFER

Table 1 evaluates the complete MICROFLOW-ST model and provides, for reference, the Stage 1a performance on long-term and short-term forecasting. The frozen MICROFLOW-ST Stage 2 check-

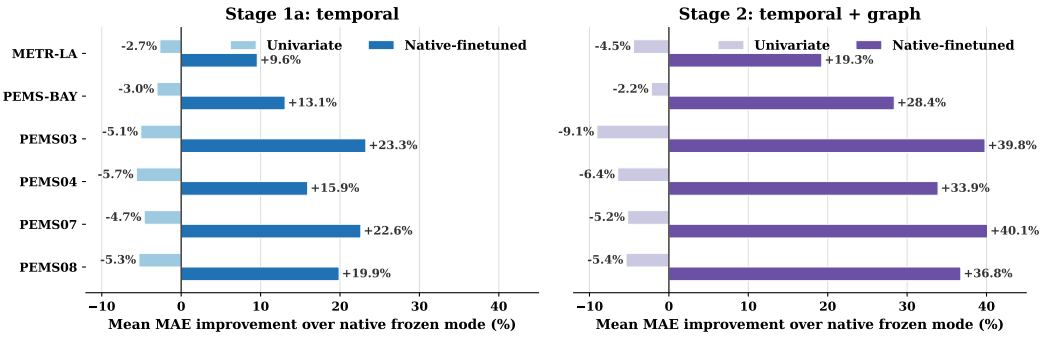


Figure 2: MAE changes relative to each stage's native frozen mode, averaged per dataset over the $12 \to 12$ and $96 \to 96$ tasks. Stage 1a multivariate mode and Stage 2 multivariate with graph mode define zero and their absolute errors appear in Table 1. Positive values indicate lower error. Stage 1a finetuning updates its forecasting head; Stage 2 finetuning updates its forecasting head and cross-modal bridge parameters.

Table 2: Staged versus scratch initialization at the same 100k Stage 2 steps. Positive deltas mean scratch has higher error. Win counts are dataset cells favoring staged initialization.

Horizon	Staged wins (MAE/RMSE)	Scratch Δ MAE	Scratch Δ RMSE
$12 \to 12$	6/6; 5/6	+4.39%	+2.64%
$96 \to 96$	4/6; 5/6	+2.05%	+3.20%
Overall	10/12; 10/12	+3.22%	+2.92%

point, pretrained without a single real system, transfers to all six unseen networks and obtains the best average MAE among the STFMs in Table 1 across datasets and horizons. At $96 \to 96$ it has the lowest MAE on five of six datasets, and it beats the published NEOST row in 8 of 12 dataset–horizon cells despite NEOST fitting a closed-form ridge map inside every input window while MICROFLOW-ST reuses one frozen checkpoint.

The Stage 1a row supplies a second reference: at $12 \to 12$ the 15.8M-parameter temporal marginal attains the lowest average MAE in the table, below models 6 to 20 times its size that were pretrained on real corpora. Graph-free multivariate context is thus already sufficient for much of the short-horizon signal; Stage 2's clearest gains over it are on the four long-horizon flow datasets.

6.3 LIGHTWEIGHT TARGET SPECIALIZATION

After establishing frozen transfer, we ask how efficiently the learned paths specialize when a small target slice is available. Figure 2 reports each horizon's MAE change relative to its own stage's frozen error (already in Table 1). Suppressing the native context path with the univariate mode raises error on average, while restricted adaptation lowers it across nearly every task. Across all 12 dataset–horizon cells, Stage 1a head adaptation reduces MAE by 17.4% on average and Stage 2 bridge-plus-head adaptation reduces it by 33.0%; at $96 \to 96$, the corresponding gains are 25.1% and 42.4%.

The same-stage Stage 2 head-only control and the full validation trajectories are reported in Appendix E. They show that unfreezing the cross-modal bridge parameters decreases validation error faster and farther under the same data and epoch budget.

6.4 DOES THE MECHANISTIC PRIOR MATTER?

MICROFLOW low-level simulation mechanism matters: Figure 3 tests this with the same 15.8M Stage 1a architecture trained either on MICROFLOW or a TempoPFN-style functional-interaction prior. In this comparison we fix the architecture, optimizer budget, and evaluation while changing the prior. At $96 \to 96$, MICROFLOW lowers the average error across the four flow datasets by 5.1% MAE and 7.8% RMSE. The functional prior is 1.9% better in MAE on the two speed datasets,

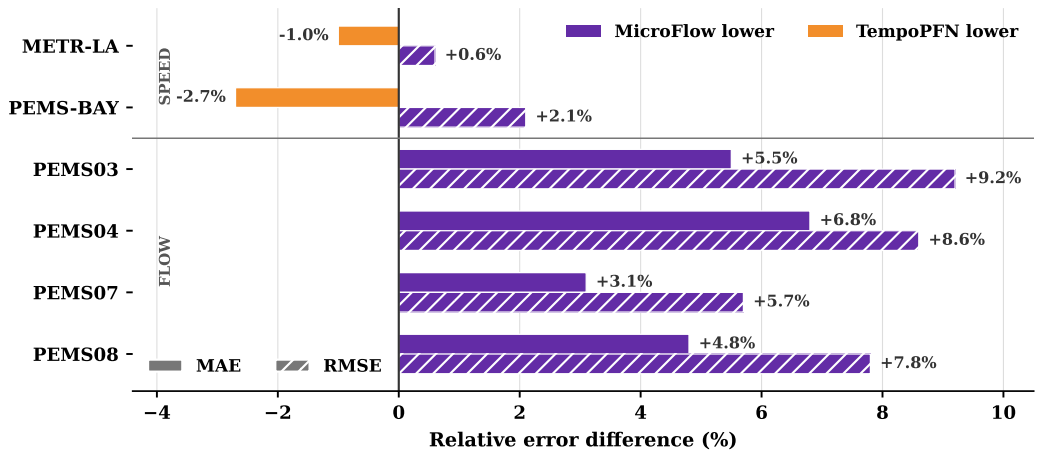


Figure 3: Matched Stage 1a prior comparison at long-term forecasting ($96 \rightarrow 96$). Bars show $100 * (\text{TempoPFN} - \text{MicroFlow})/\text{MicroFlow}$; positive values mean lower error from MICROFLOW. The architecture, training budget, and evaluation are fixed.

while MICROFLOW is 1.4% better in their RMSE. The shorter-horizon result shows the same split, but the margin between the two priors on the flow datasets is smaller: all eight flow dataset–horizon MAE cells favor MICROFLOW, whereas all four speed MAE cells favor the functional prior. The result therefore identifies where the mechanism transfers.

6.5 DOES STAGED COMPOSITION MATTER?

Finally, we train the selected complete architecture from random initialization for the same 100,000 Stage 2 steps and evaluate it under the identical frozen protocol. Staged initialization wins 10 of 12 dataset–horizon cells for both MAE and RMSE. Across all cells, scratch is 3.22% worse in MAE and 2.92% worse in RMSE; the advantage is present at both horizons (Table 2). The independently fitted marginals and initially non-perturbing cross-modal branches therefore provide a better starting point for learning the paired task.

7 CONCLUSION

In this paper we introduced two complementary ingredients for synthetic-to-real spatio-temporal transfer. Our first contribution, MICROFLOW, defines a prior over observed systems by separating constitution, marked micro-interactions, and aggregate measurement to generate more plausible spatiotemporal data. Our second contribution, MICROFLOW-ST, is an architecture that fits that prior using synchronized temporal and structural towers, graph-conditioned bridges, and a staged marginal-to-joint recipe. A single frozen synthetic-only checkpoint transfers competitively to six real-world traffic networks, outperforming many state-of-the-art spatio-tempooral and regular time-series foundation models of similar sizes. Our ablation studies further demonstrate the contributions to this performance from the mechanistic prior, the reusable coupling, and the initialization recipe used to train MICROFLOW-ST

At the same time, the MICROFLOW prior we construct focuses on physical transport and measurement relations, and the main evaluation covers traffic systems rather than general time-series or graph learning. Within that scope, the results support a concrete conclusion: generating systems before measuring series, then fitting their marginals before their coupling, is a viable route towards pretraining spatio-temporal foundation models without using data from real systems. Developing similar synthetic priors for more general spatiotemporal systems, combining effectively with real-world data, and evaluating on a broader array of forecasting tasks with diverse graph topologies are all interesting avenues for future work.

AI USE STATEMENT

Of the tasks for which ICLR requires disclosure, we used generative AI tools only to help implement methods: parts of the simulator, model, and evaluation code were written or refactored with AI assistance, and all such code was reviewed and tested by the authors. We did not use generative AI tools to generate synthetic datasets, the MICROFLOW corpus is produced by the procedural simulator described in Section 4 and Appendix B, with no LLM involvement in its generation or in any evaluation data, nor to develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, design or give feedback on research methodology or experiments, clean or reformat datasets, or interpret results. The remaining required-disclosure tasks, namely proving mathematical claims, assisting in the writing of proofs, translation, and qualitative or thematic data analysis, are not applicable to this work.

Additionally, we used generative AI tools for tasks with recommended disclosure: editing the paper to improve readability and creating or editing software code. We have reviewed all AI-assisted work and take responsibility for the final content of this paper, including any text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work raises no human-subjects concerns and releases no new real-world data. All real data used for evaluation are established public benchmarks. Our own pretraining corpus is entirely synthetic, produced by a procedural simulator from a random seed, so it contains no measurements or records about individuals and the released generator and checkpoints carry no corresponding privacy risk. The foreseeable applications are operational forecasting on infrastructure networks.

REPRODUCIBILITY STATEMENT

Appendices A and B specify the reported system completely: model dimensions, tokenization, objectives and their coefficients, the optimization schedule, and the adaptation scopes in Appendix A; the composition of the training store, the topology, dynamics, and measurement distributions, and the online task-construction transforms in Appendix B. Appendix C gives the forecasting protocol, including split boundaries and origin counts; Appendix D gives the corresponding protocol for the structural transfer experiments; Appendix E complements the forecasting ablations results.

As supplementary material we provide the MICROFLOW generator, the inference scripts for each reported checkpoint and interaction mode, and the evaluation code used to produce the reported tables. The pretraining code is not included.

The same material, together with all reported checkpoints, is available at an anonymized repository: https://anonymous.4open.science/r/microflow/README.md. The checkpoints exceed the supplementary-material size limit and are therefore only in the repository. The generated corpus is approximately 1.6 TB and is therefore not distributed; it is regenerated deterministically from the released generator and the seed recorded in Appendix B. All reported checkpoints are released. The same material will be published in a public repository on acceptance.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: A visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, 2022.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, et al. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024.

Abdul Fatir Ansari, Oleksandr Shchur, Jan Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Danielle C. Maddix, Pablo Guerron, Tony Hu, and Junming Yin. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025.

Jost Arndt, Utku Isil, Michael Detzel, Wojciech Samek, and Jackie Ma. Synthetic datasets for machine learning on spatio-temporal graphs using PDEs. *Journal of Data-centric Machine Learning Research*, 2025.

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*, 2020.

Gleb Bazhenov, Oleg Platonov, and Liudmila Prokhorenkova. GraphLand: Evaluating graph machine learning models on diverse industrial data. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Ben Cohen, Emaad Khwaja, Youssef Doubli, Salahidine Lemaachi, Chris Lettieri, Charles Masson, Hugo Miccinilli, Elise Ramé, et al. This time is different: An observability perspective on time series foundation models. In *Advances in Neural Information Processing Systems*, 2025.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha V. Naidu, and Colin White. ForecastPFN: Synthetically-trained zero-shot forecasting. In *Advances in Neural Information Processing Systems*, 2023.

Dmitry Eremeev, Oleg Platonov, Gleb Bazhenov, Artem Babenko, and Liudmila Prokhorenkova. GraphPFN: A prior-data fitted graph foundation model. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Yutong Feng, Shiyuan Piao, Yutong Xia, Xu Liu, Wenqi Fan, Fugee Tsung, See-Kiong Ng, and Yuxuan Liang. Learning spatio-temporal foundation models from pure synthetic data. *arXiv preprint arXiv:2607.16251*, 2026.

Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637:319–326, 2025.

Emaad Khwaja, Chris Lettieri, Gerald Woo, Eden Belouadah, Marc Cenac, Guillaume Jarry, Enguerrand Paquin, Xunyi Zhao, Viktoriya Zhukov, Othmane Abou-Amal, Chenghao Liu, Ameet Talwalkar, and David Asker. Toto 2.0: Time series forecasting enters the scaling era. *arXiv preprint arXiv:2605.20119*, 2026.

Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. In *Advances in Neural Information Processing Systems*, 2022.

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.

Zhonghang Li, Long Xia, Lei Shi, Yong Xu, Dawei Yin, and Chao Huang. OpenCity: Open spatio-temporal foundation models for traffic prediction. *arXiv preprint arXiv:2408.10269*, 2024.

Vladyslav Moroshan, Julien Siems, Arber Zela, Timur Carstensen, and Frank Hutter. TempoPFN: Synthetic pre-training of linear rnns for zero-shot time series forecasting. *arXiv preprint arXiv:2510.25502*, 2025.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022.

Zezhi Shao, Fei Wang, Yongjun Xu, Wei Wei, Chengqing Yu, Zhao Zhang, Di Yao, Tao Sun, Guangyin Jin, Xin Cao, Cong Gao, Christian S. Jensen, and Xueqi Cheng. Exploring progress in multivariate time series forecasting: Comprehensive benchmarking and heterogeneity analysis. *IEEE Transactions on Knowledge and Data Engineering*, 37(1):291–305, 2024. doi: 10.1109/TKDE.2024.3484454.

Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Ege Onur Taga, M. Emrullah Ildiz, and Samet Oymak. TimePFN: Effective multivariate time series forecasting with synthetic data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Proceedings of the 41st 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 *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018.

Yuan Yuan, Jingtao Ding, Chenyang Shao, Depeng Jin, and Yong Li. UniST: A prompt-empowered universal model for urban spatio-temporal prediction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Siru Zhong, Junjie Qiu, Yangyu Wu, Yiqiu Liu, Yuanpeng He, Zhongwen Rao, Bin Yang, Chenjuan Guo, Hao Xu, and Yuxuan Liang. Learning to factorize and adapt: A versatile approach toward universal spatio-temporal foundation models. *arXiv preprint arXiv:2601.12083*, 2026.

A Model and training details

A.1 Input contract and tokenization

The reported model uses hidden width $d = 256$, four attention heads, eight synchronized mega-blocks, and temporal patch size $P = 32$. A mega-block contains two within-row layers and one cross-row layer in each tower, followed by one sparse fusion layer. This section describes the actual input path used by the reported checkpoints, including the distinction between a padded batch representation and a genuine model limit.

A.1.1 Temporal rows

Let $x_{k,t}$ be sample t of series k, $m^{\text{exist}}_{k,t}$ indicate that the sample exists, and $m^{\text{vis}}_{k,t}$ indicate that it is visible to the model. The temporal scaler is causal: at every position it computes a running mean and standard deviation from visible, existing observations only. Within patch j, every position uses the running statistics at the end of that patch. If no context is visible for a row, the implementation falls back to all existing observations when constructing the scale. The standard deviation is lower-bounded by 10^{-6}. The transformed value is

$$z_{k,t} = \operatorname{asinh}\left(\frac{x_{k,t} - \mu_{k,j(t)}}{\max(\sigma_{k,j(t)}, 10^{-6})}\right), \qquad z_{k,t} \leftarrow z_{k,t} m^{\text{vis}}_{k,t}. \tag{10}$$

The inverse transformation applies $\sinh$, then restores the patchwise location and scale. asinh is approximately linear near zero but logarithmic in both tails, which reduces the leverage of extreme synthetic and real observations without imposing positivity.

For each patch, the tokenizer concatenates three length-P vectors:

$$\mathbf{r}_{k,j} = \left[\mathbf{z}_{k,j}; \neg\mathbf{m}^{\text{vis}}_{k,j}; \neg\mathbf{m}^{\text{exist}}_{k,j}\right] \in \mathbb{R}^{3P}. \tag{11}$$

The first channel contains transformed values, the second distinguishes values deliberately hidden by the forecasting objective from visible context, and the third distinguishes padding or truly missing samples from objective masking. A residual MLP with hidden width $4d$ maps $\mathbf{r}_{k,j}$ to one d-dimensional patch token. Finally, the model adds one of eight learned sampling-frequency embeddings.

The forecasting head maps each patch state directly to $P \times 9$ outputs, corresponding to the nine quantiles $\{0.1, 0.2, \ldots, 0.9\}$. All P values in a patch and all forecast quantiles are therefore produced in one forward pass; prediction is not autoregressive over the forecast horizon. Figure 4 illustrates the model architecture from input features embeddings to output heads.

A.1.2 Typed structural rows

Every structural node becomes a short row of tokens rather than one monolithic vector. For node v, with F_c continuous and F_k categorical fields, the row is

$$\mathbf{N}^0_v = [\,\mathbf{e}^{\text{id}}_v, \mathbf{e}^{\text{type}}_v, \mathbf{e}^{\text{cont}}_{v,1:F_c}, \mathbf{e}^{\text{cat}}_{v,1:F_k}\,]. \tag{12}$$

Each component has a different role:

Identifier token. A 24-dimensional randomized orthogonal feature (ORF) is projected to width d. It distinguishes nodes without assigning a corpus-wide semantic identity. The same node keeps the same identifier throughout an example, so attention can track incidence and repeated references to it.

Type token. A learned base token is combined with a learned entity, metric, or driver type embedding. Type is part of the system's constitution: it tells the model which rows can play transport, observation, or forcing roles. Driver rows and their incident control edges are removed in the reported paired forecasting stage, but the tokenizer itself supports them.

Continuous-field token. The implementation robustly standardizes each field over its visible rows in the current example. If at least four values are available, it uses

$$\widetilde{a}_{vf} = \operatorname{asinh}\left(\frac{a_{vf} - \operatorname{median}(a_{\cdot f})}{\max(1.4826\ \operatorname{MAD}(a_{\cdot f}), 10^{-3} + 10^{-3}|\operatorname{median}(a_{\cdot f})|)}\right). \tag{13}$$

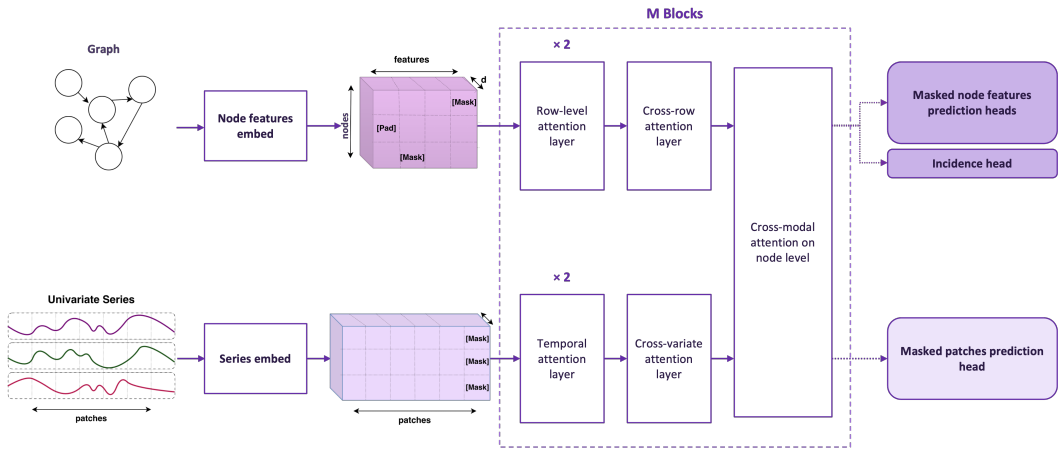


Figure 4: Detailed layout of one MICROFLOW-ST mega-block and its input and output paths. **Left:** the two modalities are tokenized independently. Node attributes become a $|V| \times F \times d$ grid of tokens, with `[Mask]` marking withheld attributes and `[Pad]` marking slots that are not fields of this example; series become a $K \times J \times d$ grid of patch tokens, with `[Mask]` marking withheld patches. **Centre:** each of the M blocks advances both towers along their two axes before exchanging information. In the structural tower, row-level attention runs within one entity across its attributes and cross-row attention runs down one attribute across entities; in the temporal tower, temporal attention runs within one series across its patches and cross-variate attention across series at the same patch position. A single sparse cross-modal layer, routed by the observed edges E and the measurement relation $\mathcal{R}$, then merges the two streams. **Right:** the active heads depend on the training stage. Stage 1b uses the masked node-attribute and incidence heads, Stage 1a the masked-patch forecasting head, and Stage 2 the forecasting head alone.

With fewer than four visible values, it uses the identity transform. The token is the sum of a shared value direction scaled by $\widetilde{a}_{vf}$, a projection of a random 32-dimensional column identifier, and a learned continuous-type embedding. Sharing the value direction exposes numerical magnitude; the independently sampled column identifier lets attention distinguish task-local fields without assigning them a global name.

Categorical-field token. A categorical value indexes one shared embedding table. The result is again combined with a random 32-dimensional column identifier and a categorical-type embedding. Consequently, code 3 in two fields need not mean the same thing: the code embedding supplies reusable categorical structure while the field identifier separates their local semantics.

Random column identifiers are redrawn for each example and shared by all rows of that example. Feature order is shuffled online. These two choices discourage position-specific meanings and make a field's values, visibility pattern, and relation to other rows the evidence from which its role must be inferred.

MASK and PAD are different operations. A masked feature is a real field whose value is deliberately withheld. Its value-dependent term is replaced by a learned MASK vector, while its column identifier and continuous/categorical type embedding remain visible. The model thus knows which field is being requested and what kind of answer it requires, but not its value. A PAD position, by contrast, is not a field in that example. It receives a shared learned padding vector (initialized at zero) and is removed from attention through the token-presence mask. Identifier and node-type tokens are never value-masked. There is no semantic "unknown category" token in the structural tokenizer: high-cardinality input fields are bucketed before tokenization, as described in Appendix D.3.

Feature slots are a runtime width, not a parameterized architectural ceiling. The checkpoint was trained with compact examples containing at most 16 continuous and 10 categorical node slots during structural pretraining, and was instantiated with 18 and 12 slots in the paired model. No

learned tensor is indexed by the slot number: every present field is tokenized independently by the same value encoder, random-identifier projection, and type embedding. The loader pads or truncates to its configured batch width, but that width can be increased when the checkpoint is instantiated without changing any learned parameter shape. Thus the trained slot counts are not a hard architectural limit, just as the number of temporal series is not fixed by a learned positional table. More fields or series increase sequence length and may constitute a distribution shift, but they do not require a new tokenizer or output layer.

The categorical vocabulary is fundamentally different. The shared categorical embedding and the categorical prediction head have 192 rows/logits in the reported checkpoint. A categorical target with more than 192 classes cannot be represented by that head without changing its parameter shape and training new class rows. At transfer time, GraphLand therefore retains the 191 most frequent input values and maps the remainder to one "other" code; it rejects a supervised categorical target whose declared cardinality exceeds 192. This is a genuine output-space limitation rather than a padding convention.

Edges contribute source and destination indices to sparse gathers but have no edge-feature token rows in the reported checkpoint. Structural reasoning therefore uses typed node attributes and incidence itself; the implementation also contains an edge-token path, but it is not part of the model evaluated in this paper.

A.1.3 Zero-initialized gated fusion

Stage 2 joins two backbones that are already useful on their own, so the cross-modal bridges must not disturb either representation at initialization. Each bridge therefore splits the keys a destination token may read into two branches: a base branch carrying the keys the structural backbone already used in Stage 1b, and a cross branch carrying the newly introduced keys of the other modality. For destination modality a at depth ℓ,

$$\mathbf{H}_a^{\ell+1} = \widetilde{\mathbf{H}}_a^{\ell} + \mathbf{O}_{a,\mathrm{base}}^{\ell} + \tanh(\alpha_a^{\ell})\, \mathbf{O}_{a,\mathrm{cross}}^{\ell}, \qquad \alpha_a^{\ell} = 0 \text{ at initialization}, \tag{14}$$

where each $\mathbf{O}$ is multi-head attention over the keys of one branch only. Because $\tanh(0) = 0$, the composed model starts out with the cross-modal branch contributing exactly nothing, so its predictions at step zero are those of the two independent backbones. The derivative of $\tanh$ at zero is one, so the gate still receives gradient and can open as training proceeds. This is the gated-residual construction used to attach new modules to frozen backbones in Flamingo and ControlNet (Alayrac et al., 2022; Zhang et al., 2023). The reported checkpoint uses one learned scalar α_a^{ℓ} per branch and block; Appendix E.3 measures what this initialization buys against training the same architecture from scratch.

A.2 Parameter allocation

The exact paired-model parameter count is 39,316,293; the temporal marginal contains 15,780,160 parameters. Table 3 reports the paired allocation by functional component. The bridge+head forecasting adaptation scope contains 8,500,816 parameters (21.6% of the paired model); head-only adaptation contains 632,384 (1.61% of the paired model and 4.01% of the temporal marginal).

Table 3: Parameter allocation of the reported MicroFlow-ST checkpoint.

Component	Parameters	Share
Temporal tower blocks	14,758,912	37.54%
Graph tower blocks	14,758,912	37.54%
Sparse fusion bridges	7,868,432	20.01%
Forecasting head	632,384	1.61%
Node prediction heads	843,253	2.14%
Tokenizers and frequency embedding	453,888	1.15%
Final stream normalizations	512	<0.01%
Total	39,316,293	100%

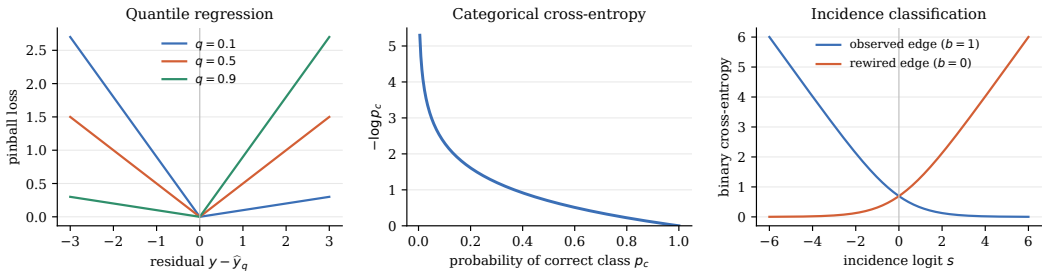


Figure 5: Probabilistic objectives used in pretraining. Left: quantile pinball loss is asymmetric around zero residual, penalizing under-prediction with slope q and over-prediction with slope $1 - q$. Middle: multiclass cross-entropy is the negative log-probability assigned to the correct class. Right: binary cross-entropy for incidence discrimination. The plots illustrate the scalar functions; training averages them only over the valid masks defined in the text.

A.3 Pretraining tasks and objectives

All pointwise losses are computed only at deliberately supervised positions and normalized by the number of valid supervised elements in the batch. The temporal objective additionally uses a count-weighted distributed reduction; structural batch losses participate in the optimizer through standard distributed gradient averaging. Normalizing before the weighted stage sum avoids giving a larger local loss merely to an example containing more nodes, fields, or valid time samples. Figure 5 visualizes the three probabilistic losses used below.

A.3.1 Quantile regression

For target y, prediction $\widehat{y}_q$, and quantile $q \in (0, 1)$, define residual $r = y - \widehat{y}_q$. The pinball loss is

$$\rho_q(r) = \max\{qr, (q-1)r\} = \begin{cases} qr, & r \geq 0 \quad \text{(under-prediction)}, \\ (q-1)r, & r < 0 \quad \text{(over-prediction)}. \end{cases} \tag{15}$$

At $q = 0.5$ this is one half of absolute error. For $q > 0.5$, failing to reach a large target costs more than overshooting it; minimizing expected pinball loss therefore estimates the conditional q-quantile rather than the conditional mean.

For forecasting, the target is scaled by the visible-history statistics and transformed with asinh exactly as in Equation 10. Let $\mathcal{I}_{\text{pred}}$ contain forecastable samples that exist but are hidden by either the suffix mask or an interior mask. The actual implementation pools both mask types into one objective:

$$\mathcal{L}_{\text{series}} = \frac{1}{9|\mathcal{I}_{\text{pred}}|} \sum_{(k,t)\in\mathcal{I}_{\text{pred}}} \sum_{q\in\{.1,.2,\ldots,.9\}} \rho_q(z_{k,t} - \widehat{z}_{k,t,q}) \,. \tag{16}$$

There is no separate coefficient for interior reconstruction in the reported runs. Suffix and interior targets differ only in how their masks are sampled, then enter the same globally normalized sum. The loss implementation can reweight fully held-out rows, but the held-out-row fraction is zero in both temporal and paired pretraining, so that branch is inactive here.

Suffix horizons are sampled uniformly from one patch through $\lfloor 0.5J \rfloor$ patches, where J is the cropped sequence length in patches. Interior corruption draws contiguous regions of one through four patches and hides each eligible region with a probability sampled uniformly between 0 and 0.25. Only forecastable rows are included in $\mathcal{I}_{\text{pred}}$.

Continuous node attributes use the same pinball definition, with quantiles $\{0.1, 0.25, 0.5, 0.75, 0.9\}$. If $\mathcal{M}_c$ is the set of masked, present continuous cells,

$$\mathcal{L}_{\text{cont}} = \frac{1}{5|\mathcal{M}_c|} \sum_{(v,f)\in\mathcal{M}_c} \sum_{q\in\{.1,.25,.5,.75,.9\}} \rho_q(a^\star_{vf} - \widehat{a}_{vf,q}), \tag{17}$$

where $a^\star_{vf}$ is the robustly transformed target from Equation 13.

A.3.2 CATEGORICAL RECONSTRUCTION

For a masked categorical cell with target class c and declared field cardinality $C_f \leq 192$, the head emits logits $\boldsymbol{\ell} \in \mathbb{R}^{192}$ but only the first C_f are valid. The remaining logits are set to $-\infty$ before the softmax. Its cross-entropy is

$$\mathrm{CE}(\boldsymbol{\ell}, c) = -\log \frac{\exp(\ell_c)}{\sum_{j=0}^{C_f-1} \exp(\ell_j)} = -\ell_c + \log \sum_{j=0}^{C_f-1} \exp(\ell_j), \tag{18}$$

and $\mathcal{L}_{\mathrm{cat}}$ is the mean over masked, present categorical cells. Restricting the normalizer matters: a binary field competes over two possible codes, not over 192 mostly inapplicable codes.

A.3.3 INCIDENCE DISCRIMINATION

The incidence task asks whether two endpoint representations are compatible under the observed system, without giving the classifier an edge-feature token. Corruption begins from observed, non-candidate edges and groups them by edge type. Within one group, it samples pairs of edges and, with equal probability, swaps their source endpoints or their destination endpoints. Because both members have the same edge type, the replacement retains the admissible endpoint-role signature; for example, a transport edge is never turned into a logically invalid driver–metric relation. The double-edge swap also preserves the total edge count and every node's in/out participation. Self-loops and duplicate edges are rejected, with at most eight resampling attempts. The requested corruption rate is 0.15; the realized rate can be lower when a small or constrained graph offers no valid swap, and the implementation records this realized fraction.

The incidence head receives only the two endpoint node rows. It first drops the identifier and type tokens, RMS-normalizes the remaining feature tokens $\{\mathbf{f}_{v,s}\}_{s=1}^{S_v}$, and pools them with a learned attention query:

$$e_{v,s} = \frac{(\mathbf{W}_k \mathbf{f}_{v,s})^\top \mathbf{q}}{\sqrt{d}}, \qquad \alpha_{v,s} = \frac{\exp(e_{v,s})}{\sum_{r \in \mathcal{P}_v} \exp(e_{v,r})}, \tag{19}$$

$$\mathbf{h}_v = \sum_{s \in \mathcal{P}_v} \alpha_{v,s} \mathbf{f}_{v,s}, \tag{20}$$

where $\mathcal{P}_v$ contains present feature slots only. For directed candidate (u, v), the score is

$$s_{uv} = \mathrm{MLP}_{\mathrm{GLU}}[\,\mathbf{h}_u; \mathbf{h}_v; \mathbf{h}_u \odot \mathbf{h}_v\,], \qquad p_{uv} = \sigma(s_{uv}). \tag{21}$$

Endpoint order is retained, so the head can learn asymmetric compatibility. All successfully faulty edges are negative examples; an equal-size random subset of uncorrupted observed edges is positive. For label $b \in \{0, 1\}$, the binary cross-entropy is

$$\mathrm{BCE}(s, b) = -b \log \sigma(s) - (1 - b) \log(1 - \sigma(s)), \quad \mathcal{L}_{\mathrm{inc}} = \frac{1}{|\mathcal{B}_{\mathrm{inc}}|} \sum_{(u,v,b) \in \mathcal{B}_{\mathrm{inc}}} \mathrm{BCE}(s_{uv}, b). \tag{22}$$

Balancing positives and negatives makes 50% accuracy the chance reference. When no valid swap is available, the code returns a differentiable zero rather than fabricating an ill-typed negative.

A.3.4 STAGE-SPECIFIC OBJECTIVES AND DISCLOSED COEFFICIENTS

Stage 1a trains the temporal tokenizer, tower, and forecast head with $\mathcal{L}_{1a} = \mathcal{L}_{\mathrm{series}}$. Stage 1b trains the structural path with

$$\mathcal{L}_{1b} = 0.5\,(\mathcal{L}_{\mathrm{cont}} + \mathcal{L}_{\mathrm{cat}}) + 0.2\mathcal{L}_{\mathrm{inc}}. \tag{23}$$

Stage 2 trains the complete composed model with $\mathcal{L}_2 = \mathcal{L}_{\mathrm{series}}$; graph reconstruction and incidence prediction are not retained as auxiliary objectives. Table 4 lists every output coefficient.

Stage 1a alternates graph-free univariate and multivariate attention masks and applies an additional univariate-view augmentation with probability 0.3. It is trained only on MICROFLOW examples. Stage 1b uses entity/transport structural views, without edge-feature tokens. Stage 2 initializes the temporal path from Stage 1a and the structural routing path from Stage 1b, replaces the earlier conditional controllers by scalar branch gates, and initializes every newly composed cross-modal output gate at $\tanh(0) = 0$. It removes exact synthetic driver nodes since those contains a 'cheating signal' because of their strong causal relationship with the system behaviour and trains the entire composed network on forecasting.

Table 4: Objective weights in the reported pretraining stages. A dash denotes a head or modality that is absent from the stage; 0 denotes an available but disabled objective.

Objective	Stage 1a	Stage 1b	Stage 2
Temporal quantile forecasting	1.0	–	1.0
Continuous node reconstruction	–	0.5	0
Categorical node reconstruction	–	0.5	0
Incidence discrimination	–	0.2	0

A.4 Optimization and learning schedule

Every pretraining stage runs for $T = 100{,}000$ optimizer steps with AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-7}$, and peak learning rate $\eta_{\max} = 10^{-3}$. Linear and convolutional weights receive weight decay 0.1; biases, normalization parameters, embedding tables, and learned base tokens receive no decay. Gradients are clipped to global Euclidean norm 1.0, and forward/backward computation uses bfloat16 mixed precision.

The warmup–stable–decay multiplier uses warmup $W = 2{,}000$, decay length $D = 20{,}000$, and stable endpoint $S = T - D = 80{,}000$:

$$g(s) = \begin{cases} s/W, & 0 \le s < W, \\ 1, & W \le s < S, \\ 1 - \sqrt{(s-S)/D}, & S \le s < T, \\ 0, & s \ge T, \end{cases} \qquad \eta(s) = \eta_{\max} g(s). \tag{24}$$

Thus the rate reaches 10^{-3} after 2,000 updates, remains exactly constant for 78,000 updates, then decays slowly at first and increasingly rapidly to zero. Figure 6 plots the exact multiplier rather than a schematic approximation.

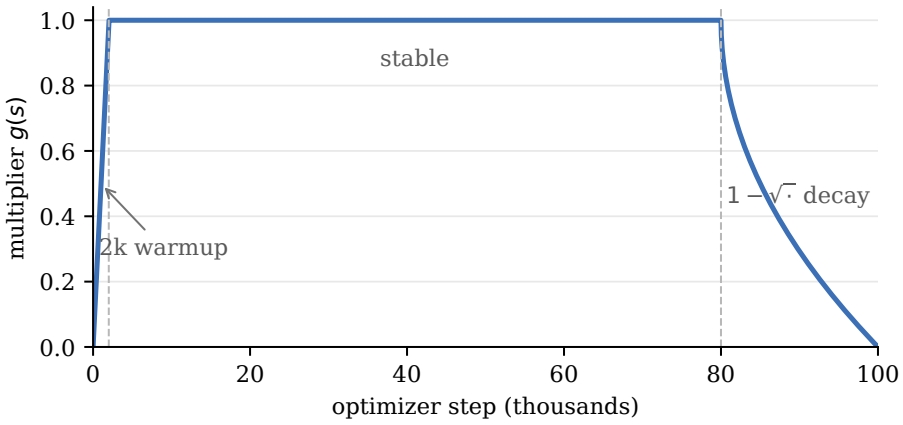


Figure 6: Exact learning-rate multiplier $g(s)$ for all three 100k-step pretraining stages.

Stage 1a uses 16 workers with per-worker batch size 2, for global batch 32, and enables gradient checkpointing. Its crop length is drawn from $\{128, 256, 512, 1024, 2048, 4096\}$ with probabilities $\{.19, .19, .19, .19, .19, .05\}$; evaluation uses 2,048 samples. Stage 1b uses 16 workers with per-worker batch size 1, for global batch 16. The packed variable-length implementation makes gradient checkpointing unnecessary in the selected Stage 1b and Stage 2 runs. Stage 2 uses 32 workers with per-worker batch size 1, returning to global batch 32.

For real-data forecasting adaptation, AdamW uses learning rate 10^{-4} for eight complete passes over origins in the final 10% of the training split. Head-only runs update 632,384 parameters. Bridge+head runs additionally update cross-modal attention and gates, for 8,500,816 parameters (21.6% of the paired checkpoint). The distinct GraphLand adaptation schedule is specified in Appendix D.6.

A.4.1 PACKED VARIABLE-LENGTH FUSION KERNEL

Graph-conditioned receptive fields have different lengths: an isolated node may expose only its own attributes, while a high-degree node may expose many incident neighbors. Padding every destination to the batch maximum would perform attention on artificial keys and allocate memory in proportion to that maximum. The implemented variable-length path instead concatenates all valid query segments and all valid key/value segments into packed tensors and records cumulative segment lengths. FlashAttention's variable-length kernel consumes these tensors directly on CUDA/bfloat16; the exact fallback loops over the same segments when that kernel is unavailable.

Packing changes neither the receptive field nor the softmax. For destination segment i with key set $\mathcal{K}_i$, it computes

$$\mathrm{Attn}(\mathbf{q}_i, \mathcal{K}_i) = \sum_{k \in \mathcal{K}_i} \frac{\exp(\mathbf{q}_i^\top \mathbf{k}/\sqrt{d})}{\sum_{r \in \mathcal{K}_i} \exp(\mathbf{q}_i^\top \mathbf{r}/\sqrt{d})} \mathbf{v}_k, \tag{25}$$

but stores only $\sum_i |\mathcal{K}_i|$ keys instead of $|\{i\}| \max_i |\mathcal{K}_i|$. When a branch has an additive group gate, its log-gate is folded into an extra key coordinate so that the packed kernel preserves the intended softmax weighting. The final paired model's scalar output gates operate after attention and require no such folding.

A.5 COMPUTATIONAL STRUCTURE AND COMPLEXITY

Let K be the number of temporal series, J their number of patches, V the number of structural nodes, $F = 2 + F_c + F_k$ the node-row length, E the number of realized edges, R the total number of routed incidence entries, M the number of mega-blocks, and d the hidden width. The reported checkpoint has $M = 8$, $d = 256$, and degree cap $\Delta = 32$ during pretraining.

Factorized temporal tower. A dense transformer over all KJ series-patch tokens would spend $\mathcal{O}(K^2J^2d)$ attention work. One within-series layer instead attends across J patches for each series, costing $\mathcal{O}(KJ^2d)$; one cross-series layer attends across K series at each patch, costing $\mathcal{O}(JK^2d)$. With two within-series layers and one cross-series layer per mega-block, attention work is

$$\mathcal{C}_T = \mathcal{O}\big(Md(2KJ^2 + JK^2)\big)\,. \tag{26}$$

The factorization preserves paths between any two temporal tokens after the alternating operations, while replacing a product of squares by a sum of axis-wise terms.

Factorized structural tower. The analogous dense cost over VF node-field tokens is $\mathcal{O}(V^2F^2d)$. Attention over fields within each node costs $\mathcal{O}(VF^2d)$; attention over nodes at a common field position costs $\mathcal{O}(FV^2d)$. The two-within/one-cross pattern gives

$$\mathcal{C}_G = \mathcal{O}\big(Md(2VF^2 + FV^2)\big)\,. \tag{27}$$

This topology-free cross-row term is why the structural marginal can also act as a tabular in-context learner: incidence changes the sparse fusion route, not the definition of a field token.

Sparse bridges. For each routed node or measurement reference, cross-modal attention combines J temporal patch positions with F structural positions. Up to constant projection factors, the realized bridge work is

$$\mathcal{C}_F = \mathcal{O}(MdJF(V + R))\,, \qquad R = \sum_{v \in V} d_v \le V\Delta, \tag{28}$$

and for a simple incidence list $R \le 2E$. A dense padded construction would allocate $\mathcal{O}(MV\Delta JF)$ even when most nodes have much smaller degree; packed storage is instead $\mathcal{O}(M(V+R)JF)$ over realized segments. The cap bounds pathological hubs while the variable-length kernel preserves the savings of sparse graphs.

Combining the three terms, the dominant attention work of the reported no-edge-token model is

$$\mathcal{O}\big(Md[2KJ^2 + JK^2 + 2VF^2 + FV^2 + JF(V+R)]\big)\,. \tag{29}$$

Token projections and feed-forward networks add $\mathcal{O}(M(KJ + VF)d^2)$ work.

The derivation also makes the two independent extensibility statements precise. Increasing the number of series K or structural fields F changes runtime and memory but not parameter shapes. Increasing the categorical class ceiling changes the embedding and classifier matrices, and is therefore not free in the same sense.

B SYNTHETIC-DATA DETAILS

B.1 THE TRAINING CORPUS

The reported checkpoints are trained on the MICROFLOW corpus described below. Each stored example is an independently sampled system, rather than another window from a small collection of long simulations. We first sample a system program, construct its graph and dynamics, simulate 256 private burn-in steps, and retain the following 4,096 steps. Table 5 summarizes the resulting store. The two entity strata receive equal mass; within a stratum, the entity count is sampled uniformly over the inclusive range. Hence large systems are not crowded out by the much larger number of possible small graphs.

Table 5: Composition of the MICROFLOW training store. Counts are numbers of independently generated systems; M denotes million and percentages are relative to the complete 4.16M-system corpus.

Split	16–48 entities	48–128 entities	Total (share)
Train	2.00M	2.00M	4.00M (96.15%)
Validation	0.04M	0.04M	0.08M (1.92%)
Test	0.04M	0.04M	0.08M (1.92%)
Total	2.08M	2.08M	4.16M

Systems are generated reproducibly from seed 42 and stored in shards of 32 (130,000 shards in total). The base sampling interval is chosen uniformly from 5 minutes, 1 hour, and 1 day. Individual metric series are then independently rolled out at stride $r \in \{1, 1, 1, 2, 3, 6, 12\}$, sampled uniformly from this multiset; the repeated unit stride retains a series at the base frequency in three sevenths of draws. Extensive measurements (counts and sums) are accumulated, while intensive measurements are averaged. The complete store occupies approximately 1.6 TB.

B.2 TOPOLOGY PROGRAMS

Each system overlays between one and four distinct topology families (Table 6). Families are drawn uniformly without replacement from the nine families in Table 7. Thus no family is privileged: because one system may contain several families, each appears in 25.3% of systems in expectation. Composition is followed by edge thinning, duplicate control, optional preferential-attachment peripheries, triadic closure, and fragmentation. These stages broaden density, degree, clustering, and connectedness without changing the family prior.

Table 6: Topology-composition distribution in the training corpus. $\mathrm{LogU}(a, b)$ denotes a log-uniform draw and $U(a, b)$ a continuous uniform draw.

Component	Distribution	Role
Number of families	$1/2/3/4$: 30/30/22/18%	multiplex complexity
Density multiplier	$\mathrm{LogU}(0.45, 3.2)$	shared density scale
Union-density target	$U(0.7, 1.8)$	pre-union edge retention
Degree correction	off 33%; else $\sigma \sim U(0.3, 1.2)$	within-block heterogeneity
Maximum duplicates	uniform in $\{1, 2\}$	overlapping family edges
Fragmentation	off 35%; else $U(0.03, 0.30)$	peripheral node isolation
PA periphery	off 50%; else fraction $U(0.10, 0.45)$	core–periphery variation
Triadic closure	off 35%; else $U(0.03, 0.50)$	clustering variation

Table 7: The nine equiprobable topology families. Ranges are conditioned on the shared density multiplier d and clipped where indicated by the implementation.

Family	Structural motif	Principal randomized controls
Chain	one or more directed corridors with branches	neighborhood k; 1–3 lines; branch probability $U(0, .35)$
Grid	regular local spatial connectivity	4/8 connectivity (equal); periodic wrapping 30%
Geometric	proximity graph in a latent plane	radius $\mathrm{clip}(U(.10, .30)d, .05, .75)$
Community	degree-corrected block model	2–7 blocks; within-/between-block rates; block skew
Scale-free	preferential-attachment core–periphery graph	attachment degree 1–12; random degree spread off 30%
Tree	branching hierarchy	fan-out 2–4; top-down orientation 70%
Hierarchical	two-level degree-corrected block hierarchy	2–4 groups and communities; three connectivity scales
Bipartite	multipartite or layered interaction system	2–4 parts; layered 60%; directed 70%; cross-part rate
Small-world	local ring with random shortcuts	degree 2–24; rewiring $U(0, .35)$; directed 50%

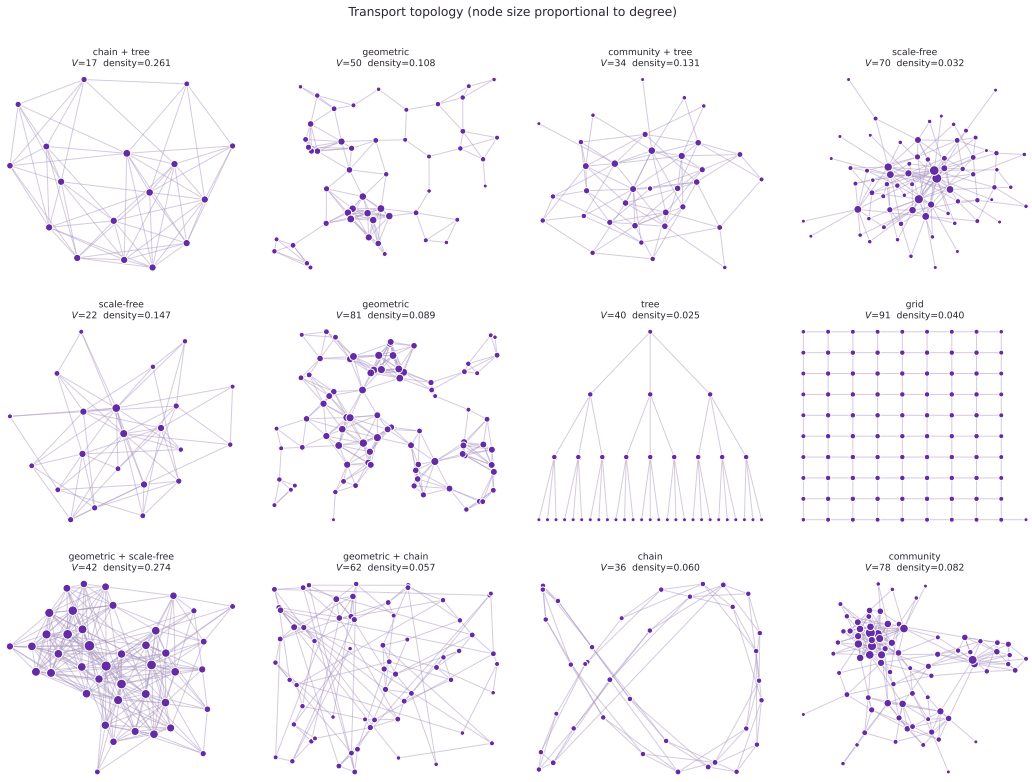


Figure 7: Representative topology programs from the training distribution, following the panel style and family coverage of the generator benchmark. Each example contains one or two topology families. We select connected realizations and use family-appropriate layouts so that the structural motif remains visible; density is directed transport-edge density.

Figure 7 shows deterministic samples from this distribution. Node size is proportional to transport degree; only entity nodes and transport edges are drawn.

Table 8: Principal dynamics and output distributions in the training corpus. Choices marked "otherwise" are conditional on the corresponding mechanism being enabled.

Axis	Distribution	Axis	Distribution
Exogenous coverage	$U(0.20, 0.50)$	Arrival-rate scale	$\mathrm{LogU}(0.15, 4.0)$
Hawkes bursts	off 50%; else $\alpha \sim U(0, 3)$	Heavy batches	on 40%; $\sigma \sim U(0.6, 1.8)$
Transit delay	on 80%; scale $U(4, 10)$	Leaky retention	on 60%; scale $U(0.3, 1.5)$
Edge capacity	on 50%; scale $U(2, 6)$	Routing preference	scale $U(0, 3)$
Temporal regime	trend 15%; seasonal 35%; default 50%	Edge operator	all pass 33%; else nonlinear share $U(.15, .70)$
Output marginal	z-score .30; positive .16; count .20		heavy .14; level .10; negative-skew .10

Table 9: Measurement and observability distribution. Statistic probabilities are the corpus mean; each system draws its own mixture from a Dirichlet distribution with concentration 1.6.

Component	Distribution	Interpretation
Instrumented entities	fraction $U(0.15, 0.85)$	partial system observability
Extra per-entity view	fraction $U(0, 0.70)$	second statistic on selected entities
Quantile view	fraction $U(0, 0.50)$	additional per-entity p90 metric
Rule metrics	none 12%; else maximum 2–13	predicate-selected group measurements
Composite metrics	none 25%; else 1–7	arithmetic combinations of base metrics
Statistic mean prior	count .28; sum .16; mean .18; std .12	p10 .09; p50 .08; p90 .09
Instrument distortion	off 30%; otherwise per-series cascade	gain jitter, noise, damping, spikes, quantization
Spike injection	35% marginal probability	rate $U(.001, .02)$; scale $U(2, 8)$
Damping / quantization	38.5% / 21% marginal probability	strength $U(.1, .8)$ / grid $U(.05, 1)$

B.3 Dynamics and temporal diversity

A system style is a continuous point $(w_{\text{transport}}, w_{\text{advection}}, w_{\text{common}}) \sim \mathrm{Dirichlet}(0.6, 0.5, 0.5)$, not a discrete family label. It jointly modulates routing, source placement, and exogenous forcing while retaining mixtures between the three vertices. Node and edge parameters combine graph-diffused latent variables with random nonlinear functions. Typed particles then move through queues, delays, capacity constraints, retention, routing, and endpoint-type-dependent interaction laws. Table 8 records the principal corpus-level draws.

The trend corner uses one additive global driver, high gain, and 60–95% edge dropout; the seasonal corner strengthens coherent periodic forcing. Driver-dominated style draws use three to six drivers. For the remaining systems, the driver library composes stochastic, periodic, autoregressive, random-walk, pulse, and regime components over multiple scales. These choices are intentionally independent of entity count and topology family.

B.4 Measurement and observability

The training measurement graph contains entity, metric, and driver nodes. A metric aggregates one or more entities and carries the resulting series; observation edges encode that association. Instrumented entities are sampled at a system-specific fraction, and additional rule and composite metrics create group and derived measurements. Table 9 gives the exact high-level sampling distribution.

The reported checkpoint uses the simulator's physical transport graph as its model-facing entity graph, augmented by the metric/entity measurement associations. Exact driver nodes and their incident control edges are removed when Stage 2 tasks are constructed; the forecasting model must



Figure 8: Empirical descriptors for 96 systems sampled directly from the training store: the first 48 records from each entity-size stratum, read from two fixed shards per stratum. Vertical lines show medians. "Inactive series" is the fraction flagged by the post-observation liveness diagnostic; "adjacent/far correlation" is the ratio of absolute contemporaneous correlation for graph-adjacent and non-local series; and "lag gain" is the improvement from optimizing a local lag on adjacent pairs.

therefore infer latent forcing from observed history and available topology. No corrupted or alternative graph relation is sampled in the corpus used for this paper.

B.5 Empirical coverage of the stored corpus

Figure 8 measures the generated systems after simulation and observation rather than merely restating their input priors. The diagnostic spans structural, univariate temporal, and graph–series coupling descriptors. In particular, the broad tails in connectivity, spectrum, marginal shape, and adjacent-series association show that a fixed prior over generator knobs does not collapse to a narrow family of realized systems.

B.6 Online task construction and data augmentation

The stored corpus fixes a simulated system and its measurements, but not a single learning problem. The data loader constructs a new view every time it visits that system. This distinction is important: the simulator supplies diversity in constitutions, interactions, trajectories, and instruments, while the online pipeline supplies diversity in schemas, context/query partitions, missingness, forecast horizons, and graph reliability. Augmentations are sampled consistently within one example—for instance, a feature warp is shared across all nodes of a graph—so the result still represents a coherent dataset rather than independent cell noise.

The transforms run in a fixed semantic order:

$$\underbrace{\text{precision} \to \text{type morph} \to \text{synthetic categories}}_{\text{define the presented schema}} \to \underbrace{\text{feature retention} \to \text{distribution warp}}_{\text{define observed covariates}} \\ \to \underbrace{\text{feature/series masks}}_{\text{define prediction targets}} \to \underbrace{\text{incidence corruption}}_{\text{perturb routing}} \to \text{slot shuffle}. \tag{30}$$

Defining the schema before masking ensures that a newly created field can itself become a scored target. Shuffling last moves its value, presence flag, mask, and cardinality together.

B.6.1 TEMPORAL VIEWS

Random crop and sampling-frequency view. From each 4,096-step stored trajectory, temporal pretraining randomly crops a length in $\{128, 256, 512, 1024, 2048, 4096\}$ with probabilities $\{.19, .19, .19, .19, .19, .05\}$. Crop location is also random. Together with the per-series stride described above, this exposes both short local patterns and long context without storing multiple copies of a system. The associated frequency code is passed to the temporal tokenizer.

Forecast suffix. For a cropped record containing J patches, the suffix length is sampled uniformly from one through $\lfloor J/2 \rfloor$ patches and hidden on forecastable rows. This is the primary forecasting task: the model observes a prefix and predicts all hidden suffix samples in parallel.

Interior contiguous-patch masking. An additional transform partitions the record into regions whose width is sampled uniformly from one through four patches. A masking probability is then drawn uniformly in $[0, 0.25)$ and applied to those regions. Contiguity prevents the task from degenerating into pointwise interpolation and teaches recovery at several temporal scales. Suffix and interior targets enter the same masked quantile objective in Equation 16; neither receives a separate loss weight.

Univariate view. With probability 0.3 in Stage 1a, cross-series temporal attention is suppressed for the batch. The same weights must therefore support both independent univariate forecasting and multivariate context. This produces an explicit graph-free temporal marginal rather than making successful prediction contingent on other channels being present.

Driver visibility. The generic pipeline can demote a known-future driver to an ordinary forecast target by clearing its known-covariate flag before temporal masks are sampled. Stage 1a applies this independently per driver with probability 0.25, allowing mixtures of known and unknown forcing in one system. The reported paired stage takes the stricter route: it removes driver nodes and their control edges entirely, so no exact future forcing is available at deployment time. Whole-series holdout is implemented by the loader but disabled in the reported Stage 1a and Stage 2 runs.

B.6.2 SCHEMA AND VALUE AUGMENTATIONS

These transforms act per node/edge role. A field owned by entities is never silently repurposed as a metric or driver field; its presence block remains confined to rows of the same type.

Precision morphing. Each present continuous slot is selected with probability 0.2. If selected, its observed range is quantized to a uniformly drawn number of levels between 4 and 64. The grid is shared by every row of that field. This exposes the model to counts, rounded coordinates, coarse sensors, and other limited-precision measurements while preserving the field's ordering and width.

Continuous-to-categorical morphing. For each role, with probability 0.3 a present continuous field is selected and quantile-bucketed into 2–6 categories. Its values and masks move into a spare categorical slot, and the original continuous slot becomes absent. This asks the same underlying information to appear under either numerical or categorical encoding.

Categorical-to-continuous morphing. Independently with probability 0.3, one categorical node field is ordinally encoded as $c/(C-1)$ in a spare continuous slot and removed from its categorical slot. This direction is used for node fields, but not for edge fields because the edge-type category is structural rather than an ordinal measurement.

Logical categorical synthesis. With probability 0.15 per role, two continuous fields are thresholded at their medians. Their two binary indicators are combined into a four-class field. Unlike an arbitrary random label, this column is predictable from visible covariates, so masked rows form a meaningful in-context task.

High-cardinality categorical synthesis. The simulator's native categorical fields contain at most a small number of codes, which would leave most rows of a 192-class embedding and classifier

untrained. The online transform therefore makes up to two additional categorical fields per role, each attempt firing with probability 0.5. It samples among five learnable constructions:

- fine quantile bins of one continuous field;
- a positional combination of coarse bins from two or three continuous fields;
- the Cartesian-product code of two existing categorical fields;
- the nearest of sampled centroids in a standardized multi-feature space; or
- quantile bins of a row's series level, spread, range, trend, or lag-one autocorrelation.

Cardinalities are drawn approximately log-uniformly and capped at 192. Combinatorial fields retain their declared vocabulary even if a small sampled graph does not exhibit every possible code, matching a real subgraph drawn from a larger schema. The construction precedes feature masking, so these columns are sometimes targets and the full classifier receives training signal.

Feature retention. Whole slots are independently retained, consistently across all rows that own them. Stage 1b keeps each node slot with probability 0.8 and guarantees at least one retained slot for every role that originally had features. Stage 2 uses probability 0.5, deliberately presenting the structural helper with sparser metadata while retaining the same minimum. Optional edge-feature slots use probability 0.8 and may all be removed; the reported checkpoint has no edge-token stream, so this edge-feature setting does not alter its representation. Dropped fields are marked absent and become PAD, not MASK: they are invisible to both attention and the reconstruction loss.

Continuous distribution warp. For each continuous node or edge slot, a warp is applied with probability 0.5. It optionally uses the signed logarithm $\operatorname{sign}(x)\log(1+|x|)$ with probability 0.2, multiplies by a log-normal scale with log standard deviation 0.6, adds a zero-mean Gaussian shift with standard deviation 0.5, optionally flips sign with probability 0.2, and clips the result to $[-8, 8]$. Parameters are shared across all rows of the field. The transform varies scale, offset, skew, and orientation without destroying the within-dataset relation the model must learn.

Feature-order shuffle. Continuous and categorical slots are independently permuted after all other transforms. Values, cardinalities, presence flags, and target masks use the same permutation. Because the tokenizer also assigns random task-local column identifiers, a persistent slot index cannot become a shortcut for a semantic field name.

B.6.3 STRUCTURAL PREDICTION VIEWS

Context/query node split. Node-feature reconstruction applies only to entity rows. A context fraction is drawn uniformly from 0.05 to 0.90 per batch; context rows keep all their fields visible, whereas query rows are eligible for masking. Within eligible rows, the cell-mask rate is drawn log-uniformly from 0.05 to 0.60, separately covering light imputation and heavily missing records. A masked cell retains its field identifier and type but replaces its value with MASK; the loss sees the original value as its target.

Sparse-label episodes. Cellwise masking does not reproduce node classification, where every covariate is observed but a single label column is visible on only some nodes. With probability 0.7 per batch, the pipeline therefore selects one live continuous or categorical field and draws its visible-label rate log-uniformly from 0.05 to 0.90. It hides that column on the remaining entity rows while leaving all other columns dense. The label modality, column position, and rate are shared by the batch. This creates both low-label and high-label in-context episodes without declaring a corpus-wide target column.

Edge-feature reconstruction view. The general loader applies the same context/query and cell-masking construction to transport-edge attributes. It keeps context edges intact and masks present fields only on query edges; if degree capping removed transport edges, it disables edge-feature targets because the surviving flow attributes no longer form a conserved system. The reported checkpoint has no edge-token stream, and its edge-feature loss is zero, so this transform creates no scored target in the selected runs.

Table 10: Principal online-view settings for the reported checkpoints. "Input only" means the transform changes what Stage 2 sees but its corresponding structural prediction loss is disabled.

Transform	Stage 1a: temporal	Stage 1b: structural	Stage 2: paired forecasting
Temporal crop	128–4096, random	inactive objective	inherited paired crop
Suffix horizon	1–50% of patches	inactive objective	1–50% of patches
Interior regions	1–4 patches; rate $U(0, .25)$	inactive objective	1–4 patches; rate $U(0, .25)$
Whole-series holdout	disabled	inactive objective	disabled
Univariate temporal view	probability .30	–	–
Driver treatment	per-row demotion .25	drivers excluded by compact view	driver nodes removed
Cell-mask rate	irrelevant to loss	$\mathrm{LogU}(.05, .60)$	input only
Context fraction	irrelevant to loss	$U(.05, .90)$	input only
Sparse-label episode	irrelevant to loss	prob. .70; rate $\mathrm{LogU}(.05, .90)$	input only
Precision/type/logical morph	irrelevant to loss	.20/.30/.15	.20/.30/.15, input only
High-cardinality synthesis	irrelevant to loss	prob. .50, at most 2/role	same, input only
Node-slot retention	irrelevant to loss	.80, minimum 1/role	.50, minimum 1/role
Distribution warp	irrelevant to loss	.50 per slot	.50 per slot
Faulty incidence	–	requested fraction .15, scored	fraction .15, input only
Candidate-edge holdout	–	disabled	fraction .15, topology dropout only
Slot shuffle	irrelevant to loss	always	always

Degree cap. The loader caps routed incidence at 32 neighbors per node during pretraining. This bounds hub cost before collation; the sparse bridge then consumes only the retained incidence. For objectives that depend on conserved edge flow, a degree-capped example is not used for edge-feature reconstruction, because removing an edge would make its endpoint flow attributes inconsistent. That safeguard is irrelevant to the reported no-edge-token checkpoint but is part of the general task constructor.

Candidate-edge prediction versus faulty-incidence prediction. The code supports two distinct perturbations. Candidate-edge prediction withholds real edges and adds role-compatible non-edges as explicitly marked candidate rows. Held-out real candidates are positive examples and sampled non-edges are negative examples; neither kind participates in message routing, so the answer cannot leak through the gather itself. The selected Stage 1b run disables this transform. Stage 2 withholds 15% of true edges but assigns zero weight to the presence head, so the active effect is input topology dropout rather than an auxiliary classification loss. Incidence corruption instead rewires 15% of observed same-type edges, keeps those edges in the sparse gather, and asks the embedding-only incidence head to identify them. The latter is the active Stage 1b task and is specified completely in Appendix A.3. Stage 2 retains the 15% corruption as training-time input noise but assigns zero weight to incidence classification.

The resulting stages do not merely see more masks of the same tensor. Stage 1a varies history, horizon, sampling scale, and availability of multivariate context; Stage 1b varies which schema is present and which field acts as a task; Stage 2 composes the two under incomplete metadata and noisy routing. The simulator and online constructor therefore address complementary forms of diversity: new systems are generated offline, while new prediction problems are generated online.

C Forecasting protocol details

The numbers of complete training origins and origins in the final-10% adaptation slice are, respectively, 23,967/2,397 (METR-LA), 36,458/3,646 (PEMS-BAY), 15,701/1,571 (PEMS03), 10,172/1,018 (PEMS04), 16,911/1,692 (PEMS07), and 10,690/1,069 (PEMS08) for $12 \to 12$. For $96 \to 96$, they are 23,799/2,380, 36,290/3,629, 15,533/1,554, 10,004/1,001, 16,743/1,675, and 10,522/1,053 in the same order. Split boundaries use integer truncation and the remainder is as-

Table 11: GraphLand RH datasets used in this paper. Edges are canonical unique edges after removing self-loops and duplicates; F_c/F_k is the number of continuous/categorical input fields before adding the target field. The test count includes labelled RH query nodes only.

Dataset	Nodes	Edges	F_c/F_k	RH test	Task	Metric
City Roads M	57,073	132,570	6/20	9,027	regression	R^2
City Roads L	142,257	279,061	6/20	30,867	regression	R^2
Twitch Views	168,114	6,797,557	0/4	42,085	regression	R^2
Tolokers-2	11,758	519,000	12/4	2,974	binary	AP
City Reviews	148,801	1,165,415	19/18	34,720	binary	AP
Artnet Exp	50,405	280,348	48/27	12,791	binary	AP
HM Categories	46,563	10,730,995	31/4	11,657	21-class	accuracy

signed to the test split. Incomplete terminal windows are discarded. Models retain their native input normalization; all predictions are inverse-transformed before the globally pooled masked MAE and RMSE are computed.

We follow the official BasicTS traffic-evaluation setup (Shao et al., 2024): the processed signal channels, chronological split ratios and integer boundaries, split-local complete windows, input and forecast lengths, zero-sentinel target mask, and raw-unit MAE and RMSE all match its dataset recipes. Published comparison rows are transcribed from their source papers rather than rerun in our pipeline. FACTOST-V2 reports its graph-agnostic temporal backbone in the zero-shot setting; NEOST fits a closed-form ridge map within each evaluation window; Neither models releases their weights or source code. Daggers flag suspected overlap between a baseline's pretraining corpus and the evaluation data. For OPENCITY, the source paper marks METR-LA, PEMS-BAY, PEMS04, and PEMS08 as present during pretraining. The disclosed MOIRAI corpus includes PEMS-BAY and PEMS03/04/07/08. Chronos-2 (Ansari et al., 2025) was pretrained on hourly and daily aggregations of LargeST's 2017 Caltrans PeMS traffic flow, creating probable underlying-observation overlap with PEMS07 and related source, location, and time exposure for PEMS-BAY; the other four evaluation periods do not overlap that 2017 slice.

D FULL GRAPHLAND COMPARISONS

D.1 PURPOSE AND SCOPE

GraphLand (Bazhenov et al., 2025) tests whether the structural marginal learned from MICROFLOW can be reused outside spatio-temporal forecasting. The checkpoint in this section is the synthetic-only Stage 1b model: it has never seen a GraphLand graph during pretraining, and the zero-shot evaluation performs no parameter update. Rather than claim that MICROFLOW-ST is a general-purpose graph learner, we use it to ask three narrow questions: can the shared field tokenizer represent unrelated real schemas, can visible labels be used in context to predict masked labels, and does supplied adjacency improve that prediction?

GraphLand defines transductive node-property tasks. We use its random high-label (RH) split, which randomly assigns approximately 50% of labelled nodes to train, 25% to validation, and 25% to test. Unlabelled nodes remain available as feature-bearing structural intermediates but never provide a target. Train labels are context; test labels are hidden queries. This is an in-context task for the frozen checkpoint and becomes supervised transductive adaptation only in the explicitly adapted row.

D.2 DATASETS AND METRICS

Table 11 describes the exact loaded artifacts. The collection spans sparse road networks and very dense interaction graphs, purely categorical through predominantly continuous schemas, regression, binary classification, and a 21-class target. It is therefore a substantial schema shift from the typed entity/metric systems used for pretraining.

For binary targets we report average precision (AP), the area under the stepwise precision–recall curve:

$$\mathrm{AP} = \sum_n (R_n - R_{n-1}) P_n, \tag{31}$$

where P_n and R_n are precision and recall after including the nth score-ranked prediction. AP emphasizes ranking positive nodes under class imbalance. Multiclass accuracy is the fraction of test nodes whose largest predicted class probability matches the target. Regression uses

$$R^2 = 1 - \frac{\sum_i (y_i - \widehat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}. \tag{32}$$

The loader standardizes regression targets using RH training statistics. Because R^2 is invariant to a shared nonzero affine transform of target and prediction, scoring in standardized or original units is identical.

D.3 Mapping a node-property task to the pretrained interface

All dataset nodes are encoded as entity rows and all dataset edges as transport incidence. Numeric fields occupy continuous slots and categorical fields occupy categorical slots. We append the task label as one additional field: a continuous field for regression and a categorical field for binary or multiclass classification. On RH training nodes its value is visible; on validation/test nodes and naturally unlabelled nodes it is replaced by MASK. Only the requested validation or test nodes enter the score, so a filler value in an unlabelled masked cell cannot affect the metric. Every ordinary covariate remains visible.

This construction turns node prediction into the same operation used during structural pretraining: infer masked values from a task-local schema and other rows that expose the same field. Binary and multiclass labels use the categorical head, with declared cardinality 2 or 21. For regression, the median ($q = 0.5$) of the continuous quantile head is inverted through the model's robust feature scaler. For binary classification, the prediction is the softmax probability of class 1; for multiclass classification, it is the softmax argmax.

The loader makes the following preprocessing choices explicit:

- numerical and fraction fields are passed in their raw representation; the checkpoint's robust median/MAD–asinh scaler performs normalization;
- surviving missing numerical values are imputed by their most frequent value after any dataset transform;
- categorical columns are deterministically factorized; if a covariate has more than 192 codes, the 191 most frequent are retained and the remainder share an "other" code;
- a target class space is never bucketed, because that would change the task; a target above the checkpoint's 192-class ceiling would be rejected;
- self-loops and duplicate edges are removed, and undirected data are canonicalized to one edge row per unordered pair. The model constructs the two endpoint incidences itself.

The runtime continuous and categorical slot widths are increased to fit each native schema plus its label field before constructing the model. As established in Appendix A.1, this does not add or reshape learned parameters because every slot uses shared tokenization and heads. No edge features are supplied.

D.4 Inference modes

The evaluation code provides four structural tiers, distinguishing adjacency from the optional edge-token stream:

`isolated_meta`. Each node is processed independently apart from attention among its own fields. This measures per-row feature use without cross-node context.

`dense_meta`. The spatial tower's cross-row layers allow nodes to attend to all other nodes at the same field position, but all graph edges are removed. This is a topology-free tabular in-context mode: visible labels can inform masked labels, but not specifically along adjacency.

adjacency_meta. This keeps the same dense structural tower and additionally activates the sparse node-to-neighbor bridge using source/destination indices. Edges are incidence only; there are no edge rows or edge features. The difference `adjacency_meta−dense_meta` therefore isolates the value of routing through actual neighbors.

graph_meta. The general implementation can also instantiate edge-token rows and node–edge exchanges. We do not report this mode: GraphLand supplies no edge attributes, and the selected checkpoint was trained without edge tokens. Constructing a stream from endpoint identifiers alone would add compute without matching the evaluated pretrained interface.

Headline results use `adjacency_meta`. We additionally report the dense/adjacency pair for adapted checkpoints in Table 14 to show what the supplied graph adds under exactly the same weights.

D.5 Zero-shot inference protocol

Whole-graph inference is infeasible because cross-row structural attention is quadratic in the number of nodes. We approximate the whole-graph context with a deterministic coverage protocol. For each evaluation seed, balanced multi-source breadth-first growth partitions the graph into chunks of at most 600 nodes. All RH training nodes inside a chunk expose their label as context, all RH test nodes inside it are queries, and every test node belongs to exactly one chunk. The reported coverage is therefore 100% by construction.

Within each induced chunk, incident degree is capped at 40. The cap first builds a randomized spanning forest with union–find, then fills remaining degree capacity with randomly ordered redundant edges. This preserves component reachability before spending capacity on cycles and avoids the node-index bias of retaining the first 40 neighbors. It cannot preserve every leaf of a hub-dominated graph, which is why the very dense Twitch, Tolokers, and HM graphs retain a smaller fraction of their global edges than the road networks.

We evaluate seeds 0, 1, and 2. A seed changes the connected partition, degree-cap realization, and random orthogonal node identifiers. Metrics are computed after concatenating predictions from all chunks for that seed, then reported as the mean across the three realizations. There is no dataset-specific search over chunk size, neighborhood size, label order, overlapping-context ensemble, or numerical transform. The frozen checkpoint receives no gradient and uses all RH training labels that co-occur with each query's partition.

D.6 Restricted adaptation protocol

Adaptation uses all RH training labels and one optimization seed (seed 0). Its scope is the node feature head plus parameters whose names belong to the repeated cross-modal fusion modules; the node tokenizer and within-/cross-row spatial backbone remain frozen. The incidence head is present in the source checkpoint but remains frozen, and no incidence auxiliary objective is used. Thus adaptation aligns the task readout and neighbor-routing path without relearning the structural representation end to end.

Training examples are connected, query-anchored neighborhoods of at most 200 nodes. Each contains up to 32 masked RH-training labels as supervised queries and up to 128 other RH-training labels as visible context. Unlabelled, validation, and test nodes may be traversed for features and structure but never expose or contribute labels. Neighborhood expansion and materialized edges both use cap 32. The packed variable-length fusion path is required.

The optimizer is Adam (not AdamW), with learning rate 5×10^{-5}, zero weight decay, and gradient-norm clipping at 1.0. Training runs for at most 2,000 updates. Every 200 updates, the checkpoint is evaluated in `adjacency_meta` on at most eight validation chunks; patience is four validation checks, and the best validation metric selects the saved checkpoint. Final evaluation returns to the zero-shot protocol: 600-node partitions, degree cap 40, 100% test coverage, and seeds 0–2. Consequently, training and evaluation differ in neighborhood size, but frozen and adapted headline values share exactly the same final evaluator.

Table 12: GraphLand RH classification results. AP is reported for binary tasks and accuracy for HM Categories; higher is better. MICROFLOW-ST results are means over three evaluation realizations. Published GFM results are transcribed from Bazhenov et al. (2025). ICL and Adapt denote frozen in-context inference and the restricted adaptation protocol, respectively. Best results are bold and second-best results are underlined.

Dataset	MICROFLOW-ST Stage 1b (ICL)	MICROFLOW-ST Stage 1b (Adapt)	OpenGraph (ICL)	AnyGraph (ICL)	TS-GNN (ICL)	GCOPE (FT)
HM Categories	<u>0.2616</u>	**0.3630**	0.1169	0.1565	0.1548	0.1999
Tolokers-2	0.3701	<u>0.4436</u>	**0.4462**	0.3021	0.3183	0.3179
City Reviews	0.5427	0.6377	0.6296	<u>0.6504</u>	0.1184	**0.6974**
Artnet Exp	0.1523	0.1839	<u>0.2372</u>	0.1580	0.1338	**0.2386**

Table 13: Complete GraphLand RH results for the synthetic-only structural checkpoint. Values use `adjacency_meta`; higher is better. "Selected step" is chosen on capped RH validation, where step 0 means that adaptation was rejected.

Dataset	Metric	Frozen zero-shot	Restricted adaptation	Change	Selected step
HM Categories	accuracy	0.2616	0.3630	+0.1014	1,800
Tolokers-2	AP	0.3701	0.4436	+0.0735	2,000
City Reviews	AP	0.5427	0.6377	+0.0950	1,600
Artnet Exp	AP	0.1523	0.1839	+0.0316	2,000
City Roads M	R^2	0.6147	0.6270	+0.0123	1,600
City Roads L	R^2	0.5993	0.6149	+0.0156	1,200
Twitch Views	R^2	0.3012	0.3003	−0.0009	0

D.7 RESULTS

Table 13 gives the complete seven-dataset result before comparing to published model families. Restricted adaptation improves six of seven tasks. The selected Twitch checkpoint is step 0: under the declared validation rule, no update beat the frozen model, so the small test difference is reevaluation variance rather than an accepted adaptation gain. The remaining datasets select checkpoints from 1,200 to 2,000 updates; Tolokers and Artnet reach the configured ceiling and should be read as budget-limited rather than converged.

The mode decomposition in Table 14 shows positive adjacency lift on every selected checkpoint, from $+0.0199$ AP on Artnet to $+0.2086$ R^2 on City Roads L. With weights fixed, routing visible labels and attributes through the supplied neighborhood substantially improves the prediction on every selected checkpoint.

Table 14: Dense versus adjacency inference for the selected adapted checkpoints under the canonical 600-node, three-realization evaluator.

Dataset	Dense	Adjacency	Lift
HM Categories	0.2375	0.3630	+0.1254
Tolokers-2	0.3784	0.4436	+0.0653
City Reviews	0.5816	0.6377	+0.0560
Artnet Exp	0.1640	0.1839	+0.0199
City Roads M	0.4370	0.6270	+0.1899
City Roads L	0.4063	0.6149	+0.2086
Twitch Views	0.1870	0.3003	+0.1133

Among the four published graph-foundation-model columns in Table 12, the frozen checkpoint exceeds 9 of 16 dataset–model entries and restricted adaptation exceeds 11 of 16. Zero-shot transfer is already competitive on HM Categories and City Reviews, while Tolokers and Artnet retain a gap to the strongest graph-pretrained comparator. Adaptation is useful on all four classification tasks but does not turn this synthetic marginal into a fully supervised specialist.

Table 15: Full GraphLand RH comparison on the four classification datasets evaluated in this work. Values are AP for binary tasks and accuracy for HM Categories. Published baseline means are from Bazhenov et al. (2025); all percentages are divided by 100. Best results are bold and second-best results are underlined.

Model	HM Categories	Tolokers-2	City Reviews	Artnet Exp
Best constant predictor	0.1946	0.2182	0.1209	0.1000
ResMLP	0.4312	0.4596	0.7521	0.4355
XGBoost	0.4668	0.4931	0.7777	0.4534
LightGBM	0.4728	0.4992	0.7788	0.4498
CatBoost	0.4698	0.5052	0.7800	0.4550
ResMLP-NFA	0.6134	0.5677	0.7980	0.4452
LightGBM-NFA	0.6709	<u>0.6219</u>	**0.8163**	0.4926
GCN	0.7338	0.6049	0.8105	<u>0.4980</u>
GraphSAGE	0.7334	0.5842	0.8075	0.4849
GAT	<u>0.7919</u>	**0.6376**	<u>0.8110</u>	**0.5062**
GT	**0.7928**	0.6032	0.8050	0.4932
OpenGraph (ICL)	0.1169	0.4462	0.6296	0.2372
AnyGraph (ICL)	0.1565	0.3021	0.6504	0.1580
TS-GNN (ICL)	0.1548	0.3183	0.1184	0.1338
GCOPE (FT)	0.1999	0.3179	0.6974	0.2386
MICROFLOW-ST Stage 1b zero-shot	0.2616	0.3701	0.5427	0.1523
MICROFLOW-ST Stage 1b adapted	0.3630	0.4436	0.6377	0.1839

Table 16: Full GraphLand RH comparison on the three regression datasets evaluated in this work. The metric is R^2 and higher is better. Published baseline means are from Bazhenov et al. (2025); all percentages are divided by 100. Best results are bold and second-best results are underlined.

Model	City Roads M	City Roads L	Twitch Views
Best constant predictor	0.0000	0.0000	0.0000
ResMLP	0.6243	0.5309	0.1336
XGBoost	<u>0.7093</u>	0.6462	0.1334
LightGBM	0.7001	0.6366	0.1337
CatBoost	0.6932	0.6124	0.1325
ResMLP-NFA	0.6587	0.5796	0.5697
LightGBM-NFA	**0.7209**	**0.6624**	0.6214
GCN	0.6995	0.6465	**0.7712**
GraphSAGE	0.7020	0.6577	0.7202
GAT	0.7053	<u>0.6603</u>	<u>0.7606</u>
GT	0.6745	0.6402	0.7557
MICROFLOW-ST Stage 1b zero-shot	0.6147	0.5993	0.3012
MICROFLOW-ST Stage 1b adapted	0.6270	0.6149	0.3003

Regression provides a complementary view because the published graph foundation models in this comparison do not expose a regression result. The frozen model reaches $R^2 = 0.6147$ and 0.5993 on the two road graphs; adaptation raises them to 0.6270 and 0.6149. Twitch is the clear negative adaptation result: the fully supervised baselines in Table 16 remain substantially stronger on Twitch.

Finally, Tables 15 and 16 include GraphLand's graph-agnostic, boosted-tree, neighborhood-feature, and supervised GNN baselines. These models are trained and tuned separately on each real dataset; the frozen MICROFLOW-ST row receives no updates, and the adapted row moves only the head and fusion modules. The comparison situates the synthetic prior's performance against dataset-specialized graph learners trained under far larger budgets.

Table 17: Stage and interaction-mode ablations. Each cell is mean test MAE / RMSE over three seeds. Univariate mode evaluates series independently; Native modes use the multivariate inference for stage 1a and multivariate with graph for stage 2. Finetuning uses the final 10% of training origins: Stage 1a updates its head, while Stage 2 updates its head and cross-modal bridges. Bold denotes the best result in each row and panel.

		(a) Stage 1a: temporal			*(b) Stage 2: temporal + graph*		
Dataset	Task	Univariate	Native	Finetuned	Univariate	Native	Finetuned
METR-LA	$12 \rightarrow 12$	4.299 / 8.513	4.176 / 8.328	**3.883 / 7.705**	4.884 / 9.765	4.483 / 8.827	**3.802 / 7.620**
PEMS-BAY	$12 \rightarrow 12$	2.342 / 5.513	2.288 / 5.418	**2.074 / 4.924**	2.734 / 6.418	2.659 / 6.284	**2.000 / 4.759**
PEMS03	$12 \rightarrow 12$	23.589 / 37.674	22.171 / 35.404	**19.382 / 31.519**	28.014 / 44.217	24.938 / 40.221	**18.431 / 29.793**
PEMS04	$12 \rightarrow 12$	30.498 / 47.168	28.891 / **44.902**	**26.853** / 45.869	35.539 / 54.071	32.650 / 51.572	**25.357 / 43.887**
PEMS07	$12 \rightarrow 12$	34.502 / 53.374	32.932 / 50.968	**29.181 / 47.199**	40.660 / 61.792	37.742 / 58.239	**27.351 / 43.964**
PEMS08	$12 \rightarrow 12$	24.741 / 39.037	23.312 / 36.954	**20.833 / 34.815**	29.008 / 45.092	26.805 / 42.590	**19.860 / 32.567**
METR-LA	$96 \rightarrow 96$	7.935 / 13.870	7.747 / 13.703	**6.806 / 12.589**	7.933 / 13.757	7.934 / 13.886	**6.084 / 11.464**
PEMS-BAY	$96 \rightarrow 96$	5.588 / 10.664	5.387 / 10.453	**4.481 / 9.094**	5.632 / 10.657	5.544 / 10.712	**3.771 / 8.140**
PEMS03	$96 \rightarrow 96$	90.906 / 128.638	87.597 / 125.439	**57.877 / 89.057**	88.959 / 127.506	84.089 / 121.300	**39.078 / 59.419**
PEMS04	$96 \rightarrow 96$	111.306 / 151.883	105.266 / 145.925	**79.162 / 124.431**	108.306 / 148.963	104.100 / 143.708	**56.775 / 86.058**
PEMS07	$96 \rightarrow 96$	133.114 / 178.959	127.322 / 173.242	**84.209 / 123.067**	128.192 / 174.644	124.889 / 170.159	**59.050 / 87.593**
PEMS08	$96 \rightarrow 96$	95.586 / 131.419	91.449 / 127.511	**64.803 / 97.853**	92.884 / 129.052	90.551 / 126.379	**47.432 / 71.549**

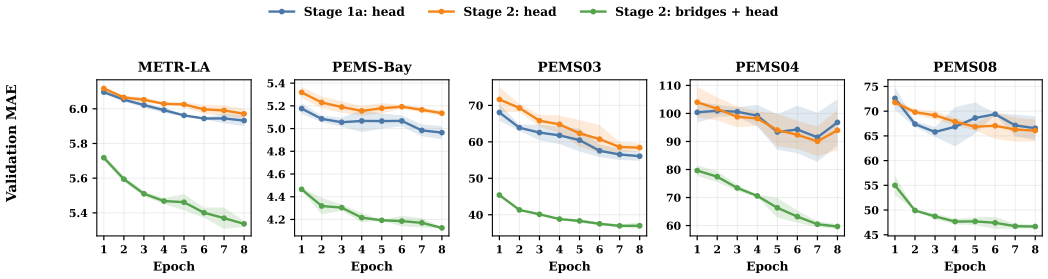


Figure 9: Long-horizon ($96 \rightarrow 96$) validation MAE during lightweight adaptation. Each panel is one dataset; points are complete passes over the final 10% of training origins and bands are sample standard deviations over three seeds. Head-only updates train 4.01% of Stage 1a or 1.61% of Stage 2; bridges+head trains 21.6% of Stage 2.

E FULL FORECASTING ABLATIONS

E.1 LIGHTWEIGHT ADAPTATION ON SPATIOTEMPORAL FORCASTING

This section gives the complete version of the adaptation experiment summarized in Section 6.3. Adaptation uses the final 10% of complete training origins for eight passes, with AdamW at learning rate 10^{-4}, no dataset-specific hyperparameter search, and checkpoint selection on the full validation split. Two scopes are compared: updating the forecast head alone (632,384 parameters, 4.01% of Stage 1a or 1.61% of Stage 2) and additionally unfreezing the cross-modal bridges (8,500,816 parameters, 21.6% of Stage 2). Everything else stays frozen.

Table 17 reports both checkpoints in three inference modes. *Univariate* evaluates each series independently; *native* uses the multivariate panel for Stage 1a and the panel together with the road graph for Stage 2; *finetuned* applies the adaptation above to the native mode. Each checkpoint's native mode is better than independent-series inference in every cell for Stage 1a, and in 11 of 12 cells on MAE and 10 of 12 on RMSE for Stage 2. The Stage 2 exceptions are confined to the long horizon: on METR-LA univariate inference is marginally stronger on both metrics, and on PEMS-BAY it is stronger on RMSE. Adaptation improves every cell on MAE. Its single regression is PEMS04 at $12 \rightarrow 12$ for Stage 1a, where MAE falls from 28.891 to 26.853 while RMSE rises from 44.902 to 45.869; since selection is on validation MAE, a small loss on the squared-error metric is possible.

Figure 9 shows the validation trajectories over all eight passes at $96 \rightarrow 96$ for five of the six datasets. Head-only updates improve both checkpoints, and the two head-only curves stay close to each other throughout, which indicates that the paired model gains little over the temporal marginal when only

Table 18: Full matched-architecture prior comparison. Values are mean raw-unit error over three inference seeds. $\Delta = 100(\text{TempoPFN} - \text{MicroFlow})/\text{MicroFlow}$, so a positive delta favors MICROFLOW. Bold marks the better prior in each cell.

Dataset	Horizon	MAE			RMSE		
		MICROFLOW	TempoPFN	Δ	MICROFLOW	TempoPFN	Δ
METR-LA	12	4.176	**4.082**	-2.3%	8.328	**8.267**	-0.7%
METR-LA	96	7.747	**7.667**	-1.0%	**13.703**	13.782	+0.6%
PEMS-BAY	12	2.288	**2.232**	-2.4%	5.418	**5.396**	-0.4%
PEMS-BAY	96	5.387	**5.240**	-2.7%	**10.453**	10.673	+2.1%
PEMS03	12	**22.171**	22.915	+3.4%	**35.404**	36.588	+3.3%
PEMS03	96	**87.597**	92.408	+5.5%	**125.439**	136.965	+9.2%
PEMS04	12	**28.891**	29.597	+2.4%	**44.902**	46.135	+2.7%
PEMS04	96	**105.266**	112.392	+6.8%	**145.925**	158.543	+8.6%
PEMS07	12	**32.932**	33.414	+1.5%	**50.968**	52.073	+2.2%
PEMS07	96	**127.322**	131.218	+3.1%	**173.242**	183.088	+5.7%
PEMS08	12	**23.312**	23.935	+2.7%	**36.954**	38.400	+3.9%
PEMS08	96	**91.449**	95.828	+4.8%	**127.511**	137.444	+7.8%

Table 19: Full staged-versus-scratch comparison at 100k Stage 2 steps. Each cell is mean test MAE/RMSE over three inference seeds. $\Delta = 100(\text{scratch} - \text{staged})/\text{staged}$; positive values favor the staged recipe. Bold marks the better initialization for each metric.

Dataset	Horizon	Scratch MAE/RMSE	Staged MAE/RMSE	Scratch Δ MAE/RMSE
METR-LA	12	4.752/9.375	**4.483/8.827**	+5.99%/+6.21%
PEMS-BAY	12	2.895/6.759	**2.659/6.284**	+8.90%/+7.56%
PEMS03	12	25.917/40.809	**24.938/40.221**	+3.93%/+1.46%
PEMS04	12	33.684/51.842	**32.650/51.572**	+3.17%/+0.52%
PEMS07	12	38.330/**57.951**	**37.742**/58.239	+1.56%/-0.49%
PEMS08	12	27.557/42.825	**26.805/42.590**	+2.81%/+0.55%
METR-LA	96	**7.747/13.778**	7.934/13.886	-2.36%/-0.77%
PEMS-BAY	96	**5.540**/10.932	5.544/**10.712**	-0.08%/+2.06%
PEMS03	96	86.220/125.616	**84.089/121.300**	+2.53%/+3.56%
PEMS04	96	108.088/150.338	**104.100/143.708**	+3.83%/+4.61%
PEMS07	96	129.932/178.703	**124.889/170.159**	+4.04%/+5.02%
PEMS08	96	94.500/132.375	**90.551/126.379**	+4.36%/+4.74%

the output scale can move. Unfreezing the bridge parameters changes the picture: the curve starts lower at the first pass and continues to fall, ending well below both head-only runs on every panel. Averaged over these five datasets, the validation-selected test checkpoints reduce Stage 2 MAE by 22.0% with the head alone and by 40.4% when the bridges are also trainable, a relative reduction $1.84\times$ larger under the same data and epoch budget; the corresponding Stage 1a head-only reduction is 23.3%. The curves are otherwise unremarkable, which is why they appear here rather than in the main text: they show monotone improvement with small seed variance and no instability, and the endpoints they select are already in Table 17.

E.2 MATCHED-PRIOR COMPARISON

Table 18 gives the absolute errors behind the relative differences plotted in Section 6.4. Both columns use the same 15.8M-parameter Stage 1a architecture, the same 100k-step optimizer schedule, and the same evaluation protocol; the only difference is the prior the model was pretrained on, MICROFLOW or a TempoPFN-style functional-interaction prior. This is the one experiment in the paper that isolates the prior.

MICROFLOW is better in 18 of the 24 dataset–horizon–metric cells. The split is not arbitrary: every cell the functional prior wins is a speed dataset, and every flow cell in the table favors MICROFLOW. The four speed MAE cells all go to the functional prior, by 1.0% to 2.7%, and it also takes short-horizon RMSE on both speed datasets by under 1%. On the flow datasets the margins are larger and grow with horizon: averaged over the four of them, MICROFLOW lowers MAE by 2.5% and RMSE by 3.0% at $12 \to 12$, and by 5.1% and 7.8% at $96 \to 96$.

E.3 STAGED VERSUS SCRATCH INITIALIZATION

Table 19 gives the per-dataset results behind the summary in Section 6.5. The scratch run uses the same architecture, the same 100,000 Stage 2 steps, the same paired task distribution, and the same frozen evaluation as the reported checkpoint; it differs only in starting from random initialization instead of the two separately pretrained backbones joined by gated no-op bridges.

Staged initialization wins 10 of 12 cells on each metric, and scratch is 3.22% worse in MAE and 2.92% worse in RMSE averaged over all cells. The advantage is larger at the short horizon, where scratch is 4.39% worse in MAE against 2.05% at the long horizon. Its four losses are concentrated in one place: long-horizon METR-LA on both metrics and long-horizon PEMS-BAY MAE, with the fourth being PEMS-BAY RMSE at the short horizon. Every flow cell favors staged initialization except PEMS07 RMSE at $12 \rightarrow 12$, where the gap is under half a percent. So the same speed-versus-flow split that appears in the matched-prior comparison (Appendix E.2) recurs here, which is consistent with both effects being about how well the prior's transport mechanism matches the target quantity rather than about the recipe alone.