SPATIAL-AS-REFINEMENT: GATED GRAPH OPERATORS OVER STRONG TEMPORAL PREDICTORS FOR SCALABLE SPATIO-TEMPORAL FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph-based spatio-temporal forecasters route information through a learned adjacency at every encoder layer, with graph operators as the substrate and temporal operators interleaved at every depth. We argue this convention is structurally fragile when the learned adjacency is noisy, as it often is on fragmented sensor networks. Under a perturbed-adjacency model, we show that substrate placement yields a noise bound that compounds multiplicatively through the operator norms of all other layers, so the optimizer cannot silence one bad layer without discarding the signal it carries. We propose **GraFT**, built on a structural reordering we call *spatial-as-refinement*: a per-sensor temporal predictor first produces a complete forecast, and a small stack of α-gated graph blocks refines it through residual updates $y \leftarrow y + \alpha \cdot \mathrm{LN}(\mathcal{G}(y))$, each gate initialized to zero. Under the same model, this reordering replaces the multiplicative bound with one that is exactly zero when gates are closed and confines noise to a channel each layer's gate controls independently. The same operator is therefore a liability or an asset depending only on where it sits, and the optimizer decides how much to use it from data alone. Experiments on controlled synthetic and real-world data confirm the theory's three predictions (substrate models diverge under adjacency noise, refinement-structured models degrade gracefully, and the mean gate closes under fragmentation) and let us verify the multiplicative-versus-gated mechanism directly. Across fourteen real-world benchmarks spanning traffic, energy, meteorology, and mobility, a single GraFT instantiation establishes new state-of-the-art performance, attaining the best average rank against strong published baselines, and the learned gates reveal where the graph actually helps.

1 INTRODUCTION

For most of the past decade, spatio-temporal forecasting has converged on a single architectural template. From STGCN (Yu et al., 2018) and DCRNN (Li et al., 2018) through AGCRN (Bai et al., 2020), GraphWaveNet (Wu et al., 2019), MTGNN (Wu et al., 2020), DGCRN (Li et al., 2023), D^{2}STGNN (Shao et al., 2022c), STAEformer (Liu et al., 2023a), and most recently MAGE (Ma et al., 2025), the leading models share a structural commitment: at every layer, representations are routed through a graph operator that mixes information across sensors. Successive papers refine what the operator computes; where it sits has not changed. We call this role the *substrate*.

A structural question. This is harmless when the adjacency is well-specified, but learned adjacencies routinely contain spurious edges, especially on fragmented sensor networks where the model invents connections that do not exist. Under a perturbed-adjacency model (Section 3), we show this noise compounds multiplicatively with every other layer's operator norm, and no scalar lets the optimizer silence a bad layer without discarding the signal it carries; this limitation is structural.

A structural alternative. One option is to remove the graph operator entirely, as PatchSTG (Fang et al., 2025) does via geography-based patching, eliminating substrate noise but forgoing relational signal. We instead construct an architecture in which graph operators can always be silenced. A per-sensor temporal predictor first produces a complete forecast, and a small stack of graph blocks refines it through gated residual updates $y \leftarrow y + \alpha \cdot \mathrm{LN}(\mathcal{G}(y))$, where $\mathcal{G}$ is a graph operator that

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.

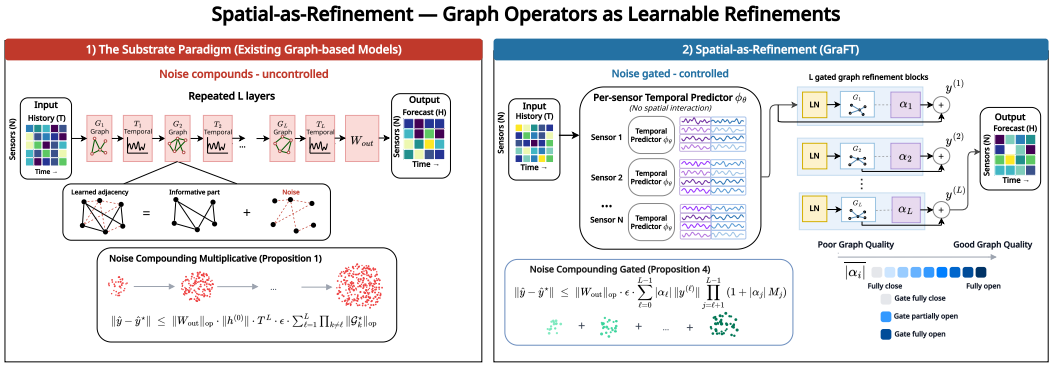

Figure 1: **Spatial-as-refinement vs. the substrate paradigm. Left (substrate):** interleaving the graph operator at every layer lets noise from the learned adjacency compound multiplicatively with depth, with no way to silence one bad layer without losing its signal (Proposition 1). **Right (GraFT):** a per-sensor predictor first produces a complete forecast; a stack of α-gated graph blocks then refines it, each gate initialized to zero, with the mean gate tracking graph quality. Noise is confined to a channel gated independently per layer, exactly zero when closed, so the optimizer can silence a bad layer without discarding the temporal baseline (Proposition 4).

mixes information across sensors, LN is layer normalization, and α is a learned scalar gate initialized to zero. We call this principle **spatial-as-refinement** and instantiate it as the **GraFT** architecture: under the same perturbed-adjacency model, this placement makes the bound exactly zero when gates are closed, with each layer's noise contribution controlled independently. The same operator is therefore a liability or an asset depending only on where it sits.

Validation by direct test of the theory. The theory makes three falsifiable predictions: substrate models degrade under adjacency noise, GraFT degrades gracefully under the same noise, and its mean gate closes as graph quality degrades. We confirm all three in a controlled synthetic task that verifies the multiplicative-versus-gated mechanism directly, and again on four real benchmarks spanning the connectivity spectrum. At full scale, we then evaluate GraFT on fourteen real-world benchmarks spanning traffic, energy, meteorology, and mobility (Section 5) against strong published baselines including MAGE and PatchSTG, across three backbones (MLP, PatchTST (Nie et al., 2023), TSMixer (Chen et al., 2023b)); the PatchTST instantiation attains the best average rank across the suite.

Our Contributions include:

1. We identify substrate-induced noise compounding: a noise bound that grows multiplicatively with depth in existing graph-based forecasters (Proposition 1).
2. We propose **spatial-as-refinement**, where graph operators contribute only through α-gated residual updates to a complete per-sensor prediction, and prove this yields a bound that is exactly zero when gates are closed, with each layer's contribution controlled independently (Proposition 4).
3. We instantiate this principle as the **GraFT** architecture, whose mean learned gate serves as a per-dataset diagnostic of graph quality.
4. We validate the theory's predictions synthetically and on real data, and achieve state-of-the-art performance against strong baselines across fourteen real-world benchmarks spanning four domains.

2 Related Work

Most spatio-temporal forecasters follow the substrate paradigm, routing the graph operator through every layer, often within a residual connection that alone provides no silencing mechanism (Appendix A): STGCN and DCRNN fuse graph convolution with temporal convolutions or recurrences; GraphWaveNet, AGCRN, MTGNN, and DGCRN make the adjacency adaptive or dynamic; D^{2}STGNN and STAEformer add decoupling or attention; and MAGE, BigST (Han et al., 2024),

and GSNet (Kong et al., 2025) target the same substrate role more efficiently. A graph-free line, including STID (Shao et al., 2022a), RPMixer (Yeh et al., 2024), and PatchSTG (spatial via patching, not adjacency), trades away relational signal for robustness on fragmented networks; GraFT keeps that signal available and lets the learned gate decide when it helps. A two-stage line instead decouples spatial and temporal computation via pretraining (STEP (Shao et al., 2022b)), structure learning (GTS (Shang et al., 2021)), error calibration (ResCAL (Kim et al., 2022)), or graph-free stacking (N-BEATS (Oreshkin et al., 2020)), yet STEP and GTS still feed a substrate GNN, ResCAL post-hoc corrects a finished forecaster's outputs with a substrate estimator driven by its past errors, and N-BEATS has no cross-sensor path; GraFT instead trains a complete forecast and its gated graph refinement jointly, refining latents with no error feedback. Gated residual methods such as ReZero (Bachlechner et al., 2021) and LayerScale (Touvron et al., 2021) gate all residual branches uniformly as optimization aids for depth, and STUGN (Bentsen et al., 2023) gates graph blocks that carry the whole spatio-temporal computation atop a persistence baseline; GraFT's gate instead selects the cross-sensor branch of an already-complete forecast, leaving a full temporal forecaster when closed and giving a direct graph-quality readout. Temporal backbones PatchTST and TSMixer, which we adopt as GraFT's predictor ϕ_θ (Section 5), offer no cross-sensor mechanism at all. Across these lines, GraFT's distinguishing combination is a complete per-sensor forecast refined by jointly trained, gated graph blocks; the full discussion is given in Appendix A.

3 METHODOLOGY

3.1 SETUP

Let $\mathcal{V} = \{1, \dots, N\}$ index N sensors, and let $x_{t,i} \in \mathbb{R}^{C_{\text{in}}}$ denote the observation at sensor i at time t. We do not assume access to a physical adjacency matrix; any spatial dependence must be learned from data. Given an input window of length T, the forecasting problem is to learn

$$f : \mathbb{R}^{T\times N\times C_{\text{in}}} \to \mathbb{R}^{H\times N\times C_{\text{out}}}, \qquad \hat{y} = f(x_{t-T+1:t}). \tag{1}$$

We call a map $\phi_\theta : \mathbb{R}^{T\times C_{\text{in}}} \to \mathbb{R}^{H\times d}$ a predictor if it operates per sensor with no cross-sensor interaction, and a linear map $\mathcal{G} : \mathbb{R}^{N\times d} \to \mathbb{R}^{N\times d}$ a graph operator if it mixes information across sensors. Throughout, $\|\cdot\|$ denotes the ℓ_∞ norm on signals (the maximum absolute entry, with the latent and horizon axes riding along the cross-sensor map), and $\|\cdot\|_{\text{op}}$ the operator norm it induces on cross-sensor maps; for a $\mathcal{G}$ with non-negative mixing weights this is the maximum absolute row sum of those weights. We write $\mathrm{LN}(\cdot)$ for layer normalization. We carry the noise analysis in ℓ_∞ because the graph operator we adopt (Section 3.4) pools every sensor through a shared, capacity-conserving bottleneck, which bounds its induced ℓ_∞ norm by a per-layer, directly computable constant (Lemma 1). Our central argument concerns neither what ϕ_θ nor what $\mathcal{G}$ computes, but where the graph operator is placed relative to the predictor.

3.2 THE SUBSTRATE PARADIGM AND MULTIPLICATIVE NOISE COMPOUNDING

Existing graph-based spatio-temporal forecasters share a structural commitment we call the **substrate paradigm**: at every layer $\ell = 1, \dots, L$, representations are mixed through a graph operator $\mathcal{G}_\ell$ interleaved with a temporal operator $\mathcal{T}_\ell$,

$$h^{(\ell)} = \mathcal{T}_\ell\Big(\mathcal{G}_\ell\Big(h^{(\ell-1)}\Big)\Big), \qquad h^{(0)} = \mathrm{Embed}(x), \qquad \hat{y} = W_{\text{out}}\, h^{(L)}. \tag{2}$$

To make the limitation mode precise, we adopt a **perturbed-adjacency model**: at each layer, the learned graph operator decomposes as

$$\mathcal{G}_\ell = \mathcal{G}_\ell^\star + \mathcal{N}_\ell, \qquad \|\mathcal{N}_\ell\|_{\text{op}} \le \epsilon, \tag{3}$$

where $\mathcal{G}_\ell^\star$ is the bounded operator induced by the informative component of the adjacency and $\mathcal{N}_\ell$ is a bounded noise operator representing spurious or uninformative edges. Let $\hat{y}^\star$ denote the prediction produced by replacing every $\mathcal{G}_\ell$ with $\mathcal{G}_\ell^\star$. To isolate the propagation of $\mathcal{N}_\ell$ through the operator product, we state the bound for the linear backbone in which the elementwise nonlinearities of $\mathcal{T}_\ell$ are treated through their operator norms; this is the linear path that the probe construction of Appendix C.4 realizes directly.

Proposition 1 (Substrate multiplicative noise bound)**.** *Assume $\mathcal{T}_\ell, \mathcal{G}_\ell^\star, W_{\text{out}}$ are bounded linear operators, and write $T = \max_k \|\mathcal{T}_k\|_{\text{op}}$, $G = \max_k \|\mathcal{G}_k^\star\|_{\text{op}}$. Then*

$$\|\hat{y} - \hat{y}^\star\| \;\le\; \|W_{\text{out}}\|_{\text{op}} \cdot \|h^{(0)}\| \cdot T^L \cdot \epsilon \cdot \sum_{\ell=1}^{L} \prod_{k \neq \ell} \|\mathcal{G}_k^\star\|_{\text{op}} \;+\; O(\epsilon^2), \tag{4}$$

whose leading term is $O\big(L\epsilon \cdot T^L G^{L-1}\big)$.

Proof sketch. The forward pass is a product of linear operators, $\hat{y} = W_{\text{out}} \prod_{\ell=L}^{1} (\mathcal{T}_\ell \mathcal{G}_\ell)\, h^{(0)}$. A first-order perturbation expansion gives $\prod_\ell (\mathcal{T}_\ell \mathcal{G}_\ell) - \prod_\ell (\mathcal{T}_\ell \mathcal{G}_\ell^\star) = \sum_\ell [\prod_{k>\ell} \mathcal{T}_k \mathcal{G}_k^\star]\, \mathcal{T}_\ell \mathcal{N}_\ell\, [\prod_{k<\ell} \mathcal{T}_k \mathcal{G}_k^\star] + O(\epsilon^2)$. Taking norms, applying $\|\mathcal{N}_\ell\|_{\text{op}} \le \epsilon$, and bounding every $\|\mathcal{T}_k\|_{\text{op}} \le T$ yields the stated bound. The full non-asymptotic statement, retaining the interleaved $\|\mathcal{T}_k\|_{\text{op}}$ factors, appears as Proposition 6 in Appendix B. □

The $O(\epsilon^2)$ remainder only adds further products of operator norms, so the multiplicative compounding is not an artifact of linearization. Two structural facts follow. **(i) Multiplicative compounding:** each summand contains a product of $L - 1$ graph operator norms, so noise at any depth is amplified through every other layer. **(ii) No silencing mechanism:** suppressing a single $\mathcal{N}_\ell$ requires also altering $\mathcal{G}_\ell^\star$, the informative component the model needs, so the substrate template forces an all-or-nothing choice per layer.

3.3 Spatial-as-Refinement and Gated Noise Bound

We propose a different structural commitment we call **spatial-as-refinement**. A per-sensor temporal predictor ϕ_θ first produces a complete forecast trajectory in latent space, computed independently per sensor:

$$y_{:,i}^{(0)} = \phi_\theta(x_{t-T+1:t,\,i},\, e_i) \in \mathbb{R}^{H \times d}, \quad i = 1, \ldots, N, \tag{5}$$

where e_i aggregates calendar and node embeddings. The collection $y^{(0)} \in \mathbb{R}^{H \times N \times d}$ is a complete prediction, computed without any cross-sensor interaction. Spatial information enters only through a sequence of L α-gated residual refinements:

$$y^{(\ell+1)} = y^{(\ell)} + \alpha_\ell \cdot \text{LN}\Big(\mathcal{G}_\ell\Big(y^{(\ell)}\Big)\Big), \qquad \ell = 0, \ldots, L-1, \tag{6}$$

where each $\alpha_\ell \in \mathbb{R}$ is a learned scalar initialized to zero. The final prediction adds a persistence skip from the last observed value:

$$\hat{y} = W_{\text{out}}\, y^{(L)} + x_{t,\,\cdot,\,\text{val}}. \tag{7}$$

The defining structural choice is that ϕ_θ alone produces a complete trajectory; $\{\mathcal{G}_\ell\}$ only adjust it. Two properties of the FFN-free recursion in equation 6 follow immediately.

Proposition 2 (Identity at initialization)**.** *With $\alpha_\ell = 0$ for all ℓ, GraFT reduces exactly to a per-sensor temporal forecaster, $\hat{y} = W_{\text{out}}\, \phi_\theta(x) + x_{t,\,\cdot,\,\text{val}}$.*

Proof. By induction on ℓ, $\alpha_\ell = 0$ implies $y^{(\ell+1)} = y^{(\ell)}$. Therefore $y^{(L)} = y^{(0)} = \phi_\theta(x)$. □

Proposition 3 (Additive decomposition)**.** *For any parameter setting,*

$$\hat{y} = \underbrace{W_{\text{out}}\, \phi_\theta(x) + x_{t,\,\cdot,\,\text{val}}}_{\textit{temporal baseline}} \;+\; \underbrace{W_{\text{out}} \sum_{\ell=0}^{L-1} \alpha_\ell \cdot \text{LN}\Big(\mathcal{G}_\ell\Big(y^{(\ell)}\Big)\Big)}_{\textit{spatial correction}}. \tag{8}$$

Proof. Unrolling the recursion gives $y^{(L)} = y^{(0)} + \sum_\ell \alpha_\ell \text{LN}(\mathcal{G}_\ell(y^{(\ell)}))$. Applying W_{out} and adding the persistence skip yields the claim. □

Proposition 2 means the optimizer starts from a strong temporal baseline, so any α_ℓ must earn its keep by reducing the loss. Proposition 3 makes the contribution directly inspectable: $\|\alpha_\ell \cdot \text{LN}(\mathcal{G}_\ell(y^{(\ell)}))\|$ measures how much layer ℓ's refinement contributed. As in Section 3.2, the bound below is stated for the linear refinement path (the operator product without LayerNorm).

Proposition 4 (Refinement gated noise bound)**.** *Under the perturbed-adjacency model $\mathcal{G}_\ell = \mathcal{G}_\ell^\star + \mathcal{N}_\ell$ with $\|\mathcal{N}_\ell\|_{\mathrm{op}} \le \epsilon$, for all $\epsilon \ge 0$, the noise contribution to the (LayerNorm-free) GraFT prediction satisfies the bound*

$$\|\hat{y} - \hat{y}^\star\| \;\le\; \|W_{\mathrm{out}}\|_{\mathrm{op}} \cdot \epsilon \cdot \sum_{\ell=0}^{L-1} |\alpha_\ell| \, \|y^{(\ell)}\| \prod_{j=\ell+1}^{L-1} \left(1 + |\alpha_j| \, \|\mathcal{G}_j^\star\|_{\mathrm{op}}\right). \tag{9}$$

Two corollaries follow:

- *(a)* ***Zero contamination when gates are closed:*** *if $\alpha_\ell = 0$ for all ℓ, the bound is exactly 0.*
- *(b)* ***Small-gate regime (near initialization, not the trained operating point; see Section 5.3 for trained magnitudes):*** *each $\mathcal{G}_j^\star$ satisfies $\|\mathcal{G}_j^\star\|_{\mathrm{op}} \le M_j$ for a per-layer, directly computable constant M_j (Lemma 1); if all gates satisfy $|\alpha_\ell| \le \alpha_{\max} \ll 1/M_j$, the product factors collapse to $1 + O(\alpha_{\max})$ and $\|\hat{y} - \hat{y}^\star\| \le \|W_{\mathrm{out}}\|_{\mathrm{op}} \cdot \epsilon \cdot \sum_{\ell=0}^{L-1} |\alpha_\ell| \, \|y^{(\ell)}\| + O(\alpha_{\max}^2)$.*

Proof sketch. Writing $\delta_\ell = \|y^{(\ell)} - y^{(\ell)\star}\|$, the triangle inequality gives the exact recursion $\delta_{\ell+1} \le \delta_\ell(1 + |\alpha_\ell| \, \|\mathcal{G}_\ell^\star\|_{\mathrm{op}}) + |\alpha_\ell| \, \epsilon \, \|y^{(\ell)}\|$, which holds at all orders in ϵ (unlike the first-order substrate bound of Proposition 1). Unrolling from $\delta_0 = 0$ gives equation 9; corollary (a) follows by setting $\alpha_\ell = 0$, corollary (b) by bounding $\|\mathcal{G}_j^\star\|_{\mathrm{op}} \le M_j$ (Lemma 1). Full derivation and proof: Proposition 7, Appendix B. □

The deployed block (Section 3.4) wraps each operator in LayerNorm, replacing $\|\mathcal{G}_\ell^\star\|_{\mathrm{op}}$ with a Lipschitz constant; the placement argument is unchanged: contamination scales with the gates and vanishes when they close. Comparing the two bounds isolates the structural difference: the substrate bound scales with $\prod_{k \ne \ell} \|\mathcal{G}_k^\star\|_{\mathrm{op}}$, a product of fixed operator norms with no scalar that can shrink it without removing informative signal, while every factor in the refinement bound's $|\alpha_\ell| \cdot \prod_{j>\ell}(1 + |\alpha_j| M_j)$ carries its own independent, trainable gate: setting $\alpha_\ell = 0$ removes layer ℓ's contribution exactly, without needing any other gate to move.

The bound above is a theoretical guarantee; we empirically verify, in Sections 4 and 5, that gradient descent shrinks α_ℓ as $\mathcal{G}_\ell$ becomes less informative.

3.4 The GraFT Architecture

We instantiate the spatial-as-refinement principle as a complete architecture. None of Propositions 2–4 depends on the internal form of ϕ_θ; they require only that it be a per-sensor predictor producing a complete trajectory. The principle is therefore backbone-agnostic, and we evaluate three instantiations of ϕ_θ in Section 5.

Per-sensor temporal predictor. The predictor ϕ_θ is any per-sensor map applied independently across nodes with weights shared across the network. Our default instantiation is a three-layer MLP over the flattened input window,

$$\phi_\theta(x_i, e_i) = W_3 \, \sigma(W_2 \, \sigma(W_1[\, x_i \,;\, e_{\mathrm{tod}} \,;\, e_{\mathrm{dow}} \,;\, e_{\mathrm{node},i} \,])), \tag{10}$$

with GELU activations σ and time-of-day, day-of-week, and per-node embeddings. We additionally evaluate PatchTST (transformer-based) and TSMixer (MLP-based) backbones in place of the MLP; the refinement stack is identical across all three, isolating the effect of the predictor. Both backbones follow designs made available through the Time-Series-Library (Wu et al., 2023); full implementation details are given in Appendix F.

Linear adaptive graph operator. The refinement step must stay bounded, since the bound of Proposition 4 controls contamination only through the gate α_ℓ and the operator norm $\|\mathcal{G}_\ell^\star\|_{\mathrm{op}}$. We therefore route every sensor through a shared, capacity-conserving bottleneck of $d_G \ll N$ latent spatial modes,

$$[\mathcal{G}_\ell(y)]_i = \sum_{j=1}^{N} w_{ij}^{(\ell)} \, y_j, \qquad w_{ij}^{(\ell)} = \sum_{k=1}^{d_G} p_{ik}^{(\ell)} \, q_{jk}^{(\ell)} \ge 0, \tag{11}$$

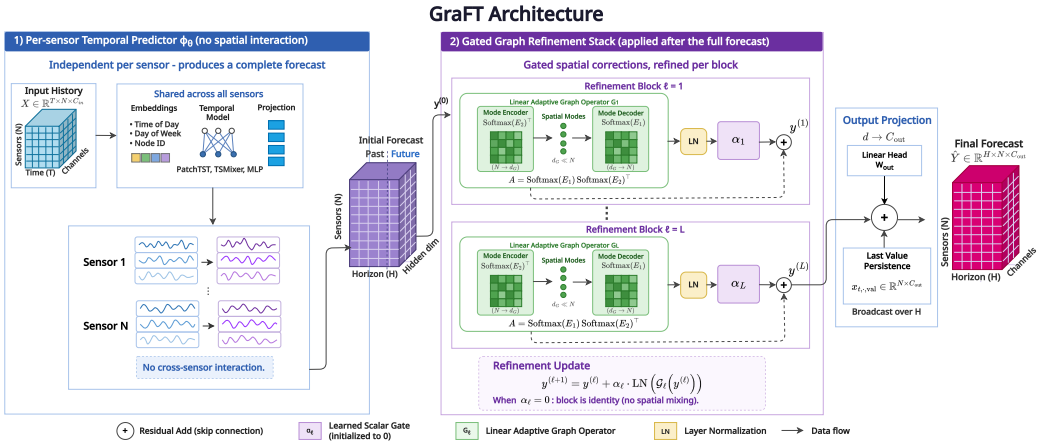

Figure 2: **The GraFT architecture.** A per-sensor temporal predictor ϕ_θ produces a complete forecast $y^{(0)}$ with no cross-sensor interaction; a stack of L α-gated graph blocks refines it, each gate initialized to zero; a linear head with a persistence skip gives the final forecast (Proposition 4 analyzes the LayerNorm-free path; full details in Section 3.4).

where $p^{(\ell)}_{i,:} = \mathrm{softmax}(E^{(\ell)}_{1,\,i,:})$ and $q^{(\ell)}_{j,:} = \mathrm{softmax}(E^{(\ell)}_{2,\,j,:})$ are each a distribution over the d_G modes, with factor matrices $E^{(\ell)}_1, E^{(\ell)}_2 \in \mathbb{R}^{N\times d_G}$. Because every sensor spends exactly one unit of routing mass, the total load absorbed by mode k,

$$c^{(\ell)}_k := \sum_{j=1}^{N} q^{(\ell)}_{jk}, \qquad \sum_{k=1}^{d_G} c^{(\ell)}_k = N, \tag{12}$$

is conserved in aggregate: nothing is created or destroyed by the pooling step, only redistributed across modes. The adjacency is never materialized; the operator is evaluated by associativity,

$$\mathcal{G}_\ell(y) = \mathrm{softmax}\Big(E^{(\ell)}_1\Big)\left(\mathrm{softmax}\Big(E^{(\ell)}_2\Big)^{\top} y\right). \tag{13}$$

Lemma 1 (Bounded mixing)**.** *Let $M_\ell := \max_k c^{(\ell)}_k$. The operator in equation 13 satisfies $\|\mathcal{G}_\ell(y)\|_\infty \le M_\ell \,\|y\|_\infty$, and its induced ℓ_∞ operator norm satisfies $\|\mathcal{G}_\ell\|_{\mathrm{op}} \le M_\ell \le N$.*

Proof sketch. Since $w^{(\ell)}_{ij} \ge 0$ and $\sum_j w^{(\ell)}_{ij} = \sum_k p^{(\ell)}_{ik}\, c^{(\ell)}_k$ is a convex combination of $\{c^{(\ell)}_k\}_k$, every row sum is at most M_ℓ, giving $\|\mathcal{G}_\ell\|_{\mathrm{op}} \le M_\ell \le N$. The full derivation appears in Appendix B. □

Lemma 1 is what makes the refinement role safe and is the reason we carry the analysis in ℓ_∞: the operator-norm factors $\|\mathcal{G}^\star_j\|_{\mathrm{op}}$ in the bound in equation 9 are bounded by the mode load M_j, a per-layer, directly computable constant, so the product $\prod_{j>\ell}(1+|\alpha_j|\,\|\mathcal{G}^\star_j\|_{\mathrm{op}})$ is controlled entirely by the gates. A refinement step cannot amplify the trajectory without limit; it redistributes it across sensors through the shared modes, gated by α_ℓ.

Remark 1. *The form in equation 13 normalizes both embeddings across modes; MAGE's kernel-trick factorization normalizes the second across nodes instead, so the two are reached differently: MAGE treats its row-stochastic form as a low-rank approximation to a full-rank adaptive graph, while we derive ours bottom-up from the capacity-conserving bottleneck requirement itself (the conservation identity underlying Lemma 1), so ours is mathematically derived, not an approximation.*

Proposition 5 (Linear complexity)**.** *Evaluating $\mathcal{G}_\ell$ on $y \in \mathbb{R}^{B\times H\times N\times d}$ via the associative form in equation 13 requires $O(BHNd_Gd)$ time and $O(Nd_G + BHNd)$ memory; no intermediate scales with N^2. Empirical training speed and maximum batch size are reported in Appendix H.*

Proof. Forming the encode $z = \mathrm{softmax}(E_2)^{\top} y$ has shape $B\times H\times d_G\times d$ at cost $O(BHNd_Gd)$. The subsequent decode $\mathrm{softmax}(E_1)\,z$ has the same cost. Both factor matrices are $N\times d_G$, giving $O(Nd_G)$ parameter memory; the largest activation is y itself at $O(BHNd)$. □

Refinement block. Each block applies the α-gated residual update, optionally followed by a post-graph FFN:

$$y^{(\ell+1)} = y^{(\ell)} + \alpha_\ell \cdot \mathrm{LN}\Big(\mathcal{G}_\ell(y^{(\ell)})\Big), \tag{14}$$

$$y^{(\ell+1)} \leftarrow y^{(\ell+1)} + \mathrm{FFN}_\ell\Big(\mathrm{LN}(y^{(\ell+1)})\Big). \tag{15}$$

The FFN is initialized following standard residual conventions so that the entire block is identity at initialization. We tune $L \in \{1, 2, 3\}$ per dataset.

Output head. A linear projection W_{out} maps the latent dimension d to the output channels, with a persistence skip from the last observed value: $\hat{y} = W_{\text{out}}\, y^{(L)} + x_{t,\cdot,\text{val}}$.

3.5 Testable Predictions

Propositions 1 and 4 yield three falsifiable predictions: **P1 (Substrate divergence).** As graph quality degrades (smaller ρ, larger effective $\|\mathcal{N}_\ell\|_{\text{op}}$), the substrate noise term grows as $O(\epsilon \cdot T^L G^{L-1})$, so substrate error stays high and is largely insensitive to whether the graph is informative, since the noise channel dominates regardless. **P2 (Refinement graceful degradation).** Under the same sweep, the refinement noise term is bounded by $O\Big(\epsilon \cdot \sum_\ell |\alpha_\ell|\, \|y^{(\ell)}\| \prod_{j>\ell}(1 + |\alpha_j| M_j)\Big)$, so GraFT degrades gracefully: as the graph fragments and the gates close, it falls back toward its own gates-closed temporal baseline (Proposition 2). **P3 (Gate closing).** For the refinement model trained under fragmentation, the mean learned $|\alpha_\ell|$ decreases monotonically as fragmentation increases, because reducing the gate is the only mechanism in the architecture that can suppress the noise contribution from $\mathcal{N}_\ell$. Predictions P1–P3 are tested under controlled conditions in Section 4 and re-examined on real data in Sections 5.2 and 5.3.

4 Synthetic Verification

The synthetic experiments are diagnostic, not evaluative: under controlled conditions that remove confounders, they test predictions P1–P3 and the noise-propagation mechanism of Propositions 1 and 4 directly (Figure 3). The goal is not to show that a graph helps (Section 5 does that) but to isolate the consequence of where the graph is placed, holding operator and predictor fixed.

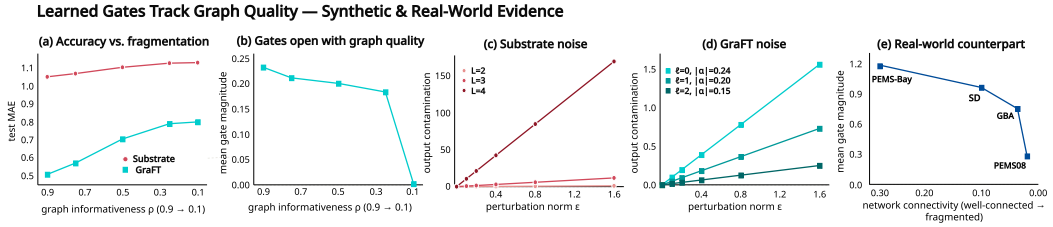

Figure 3: The learned gates track graph quality across three lines of evidence. (a)–(b) Synthetic sweep (Section 4): substrate MAE stays high and flat across ρ while GraFT degrades gracefully (P1, P2), and the mean gate closes as $\rho \to 0$ (P3). (c)–(d) Mechanistic verification (Appendix C.4): injected noise explodes with substrate depth L (c) but is ordered by each layer's trained gate $|\alpha_\ell|$ under GraFT, with no amplification through the stack (d), the multiplicative-versus-gated contrast of Propositions 1 and 4. (e) The real-world counterpart (Section 5.3): mean gate magnitude falls monotonically with connectivity across PEMS-Bay, SD, GBA, and PEMS08.

Setup. We generate $N = 200$ sensors in $K = 4$ communities, each carrying a forecastable shared latent $s_k(t)$ observable only through a near-Nyquist carrier that makes each sensor near-white on its own, while community averaging recovers s_k, so spatial mixing is genuinely irreplaceable. A parameter $\rho \in [0, 1]$ controls graph informativeness, mapping onto the effective noise norm $\|\mathcal{N}_\ell\|_{\text{op}}$ of Propositions 1 and 4. We sweep $\rho \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$, comparing two architectures at matched budgets that share the same predictor and operator: **Substrate** (operator at every layer, no gating) and **GraFT**; every difference isolates placement and gating. (Full construction including data generation, model comparison and experimental results can be found in Appendix C).

Behavioral predictions (Figure 3a–b). The substrate is strictly dominated: its MAE stays high and barely responds to ρ, routed entirely through a noise-compounding channel the architecture cannot suppress (P1). GraFT, on the identical operator, degrades gracefully as $\rho \to 0$, falling back toward the temporal baseline rather than inflating (P2). The mean gate $\overline{|\alpha_\ell|}$ tracks graph quality, closing as the graph fragments (P3) (a learned diagnostic, not a tuned hyperparameter). The gap between the two curves is the clearest illustration of the spatial-as-refinement claim: the same operator is a liability or an asset depending only on where it sits.

Mechanistic verification (Figure 3c–d). The above confirms the predicted consequences; to confirm the mechanism directly, we inject a perturbation of known norm ϵ into a linear probe path (temporal operators set to identity, LayerNorm and FFN removed) that realizes exactly the operator products Propositions 1 and 4 bound. Substrate contamination grows linearly in ϵ but explodes with depth (the multiplicative $\prod_k \|\mathcal{G}_k^\star\|_{\mathrm{op}}$ scaling) whereas GraFT contamination is linear in ϵ and ordered by each layer's trained gate, with no amplification through the stack (the gated bound of Proposition 4). This multiplicative-versus-gated contrast is the load-bearing empirical statement of the paper's theoretical claims; full sweeps and exact values are in Appendix C, and Section 5 tests whether the mechanism persists under real-world conditions.

5 Real-World Evaluation

We evaluate GraFT across fourteen spatio-temporal benchmarks spanning four domains: traffic (the four LargeST splits (Liu et al., 2023b), including SD: 716 sensors, GBA: 2,352, GLA: 3,834, CA: 8,600, the PEMS series PEMS03/04/07/08 (Song et al., 2020), and PEMS-Bay (Li et al., 2018)), energy (UrbanEV (Li et al., 2025), Electricity (Trindade, 2015)), meteorology (KnowAir (Wang et al., 2020), China City Air Quality (Chen et al., 2023a)), and mobility (Shanghai Mobile (Li et al., 2022)). The selection spans network sizes from under 200 to over 8,000 sensors and graph connectivity from well-connected metropolitan networks to severely fragmented highway corridors. Full dataset descriptions are in Appendix E.

5.1 Setup

All datasets are split chronologically 6:2:2 into train/validation/test, with a 12-step input/horizon for traffic and 24-step for the cross-domain benchmarks; performance is measured by MAE, RMSE, and MAPE (full preprocessing process and hyperparameters in Appendix F). We compare against the strongest published baselines: classical GNNs (DGCRN, AGCRN, GraphWaveNet, MTGNN), two-stage methods (GTS), linear/MLP methods (STID), efficient adaptive-graph methods (BigST), transformer-based methods (STWave (Fang et al., 2023), STAEformer, D^{2}STGNN, PatchSTG), and mixture-of-experts method (MAGE). Each baseline is run from its official implementation with the hyperparameters and training settings published by its authors. The principle is backbone-agnostic, so we report three instantiations of ϕ_θ (MLP, transformer-based PatchTST and MLP-based TSMixer) with the same refinement stack across all three. Full training implementation is in Appendix F.

5.2 Results

Table 1 reports MAE across all fourteen benchmarks; full RMSE and MAPE results appear in Appendix G. A single GraFT instantiation leads on the majority of benchmarks, and the best instantiation (PatchTST backbone) achieves the lowest average rank (1.50) across the suite, with eight outright wins.

Across LargeST, GraFT achieves the best MAE on GBA (against PatchSTG's adjacency-free 19.67) and CA, is within 0.04 of PatchSTG on GLA, and trails MAGE on SD (16.64 vs. 16.35). On the PEMS series, GraFT achieves the best MAE on PEMS03, PEMS04, PEMS07, and PEMS-Bay, and is second on PEMS08 (13.79, 0.05 behind STWave's 13.74).

Across the cross-domain benchmarks, GraFT consistently leads on Shanghai Mobile, KnowAir, China City Air Quality, UrbanEV, and Electricity. The largest gain over the best baseline is on Electricity, where GraFT (PatchTST) reaches MAE 172.4 against MAGE's 181.0 ($\sim$5% reduction). The three GraFT backbones take the top three average ranks (1.50, 2.21, and 4.29; the best baseline, MAGE, has 4.43), so the gains do not hinge on one predictor.

Table 1: **Main results (MAE, mean of 5 seeds). Best** (red) and second best (blue) per benchmark; " / "= not reported due to missing latitude/longitude metadata; "–" = out-of-memory even at the minimum batch size. #Wins/AvgRank are computed over all benchmarks (lower rank is better). The three GraFT rows share identical refinement stack, differing only in ϕ_θ. Full table in Appendix G.

Model	LargeST				PEMS					Mob.	Meteo.		Energy		#Wins	AvgRank
	SD	GLA	GBA	CA	03	04	07	08	Bay	ShMob	KnowAir	ChinaAir	UrbanEV	Elec		
GraFT (PatchTST)	16.64	19.23	**19.09**	**17.78**	**14.34**	**18.14**	**18.90**	13.79	**1.54**	0.386	15.58	17.94	**3.81**	**172.4**	**8**	**1.50**
GraFT (MLP)	16.72	19.37	19.51	17.87	14.53	18.20	19.22	13.87	1.57	**0.383**	15.46	**17.88**	3.97	177.8	2	2.21
GraFT (TSMixer)	17.02	20.12	20.35	18.74	14.87	18.38	19.54	14.26	1.58	0.386	**15.20**	17.97	3.94	182.2	1	4.29
MAGE	**16.35**	19.74	19.84	18.28	14.66	18.25	19.63	13.93	1.59	0.395	16.65	18.86	3.88	181.0	1	4.43
PatchSTG	16.98	**19.19**	19.67	17.99	/	/	/	/	1.61	0.389	16.54	19.06	4.74	/	1	5.44
STAEformer	19.02	–	21.30	–	15.27	18.78	20.09	14.17	1.65	–	16.49	19.01	4.61	200.5	0	9.55
GWNet	18.07	20.37	20.83	19.75	14.76	18.81	19.92	14.20	1.61	0.398	15.80	18.74	4.50	200.3	0	7.64
DGCRN	17.79	–	20.53	–	15.09	18.68	20.24	14.34	1.64	–	21.11	21.87	7.09	250.3	0	10.73
D^2STGNN	17.13	–	21.13	–	14.84	18.61	20.33	14.36	1.62	–	15.96	18.82	4.31	233.2	0	7.91
STWave	17.64	20.22	20.56	20.67	14.89	18.69	20.11	**13.74**	1.65	–	16.46	20.26	4.30	190.1	1	7.54
BigST	17.68	20.98	21.15	19.32	15.30	18.42	20.31	14.19	1.65	0.394	17.02	18.67	4.78	187.0	0	8.71
STID	18.03	20.40	20.65	19.04	15.37	18.29	19.54	14.19	1.70	0.393	16.68	20.54	4.42	188.9	0	8.57
AGCRN	18.39	20.26	20.69	–	15.60	19.25	20.40	15.54	1.65	–	17.53	19.57	8.39	211.5	0	12.08
MTGNN	18.21	21.75	21.48	19.91	15.31	19.20	20.97	15.22	1.63	0.394	16.08	19.62	4.74	201.3	0	10.64
GTS	18.34	–	22.06	–	14.99	20.33	21.34	15.40	1.67	0.405	17.66	19.82	4.41	241.1	0	12.25

5.3 Ablations and Learned Gates on Real Data

Setup. We ablate the refinement and its α-gate on four benchmarks spanning the connectivity spectrum (defined in Appendix D): *no graph* (zero refinement layers, the pure predictor of Proposition 2), *no gating* ($\alpha_\ell = 1$, a refinement-shaped substrate that cannot silence a layer), and *identity graph* ($\mathcal{G}(y) = y$: gate, LayerNorm, and feed-forward layer kept, no cross-sensor mixing).

Table 2: **Ablations across the connectivity spectrum (MAE, mean of 5 seeds, PatchTST instantiation).** For each benchmark we report network connectivity, the trained mean gate magnitude $\overline{|\alpha_\ell|}$, the full GraFT model, and three ablations: *no graph*, *no gating*, and *identity graph*. $\Delta\%$ columns give the MAE increase over full GraFT. Datasets are ordered from most to least connected.

Dataset	Connectivity	$\overline{\lvert\alpha_\ell\rvert}$	GraFT	No graph		No gating		Identity graph	
				MAE	$\Delta\%$	MAE	$\Delta\%$	MAE	$\Delta\%$
PEMS-Bay	0.280	1.18	1.54	1.60	3.9	1.58	2.6	1.59	3.2
SD	0.101	0.96	16.64	17.61	5.8	17.11	2.8	17.39	4.5
GBA	0.037	0.75	19.09	20.06	5.1	19.47	2.0	19.78	3.6
PEMS08	0.017	0.28	13.79	14.28	3.6	14.00	1.5	14.08	2.1

Results. As shown in Table 2, removing the graph degrades MAE on every benchmark, so the graph carries signal the per-sensor predictor cannot recover. The identity graph, which keeps the block's capacity, recovers only 17–41% of that gap, so most of the gain comes from learned cross-sensor mixing, not added per-sensor capacity. Fixing the gates at $\alpha_\ell = 1$ is never free either, though its cost tracks connectivity less cleanly than the trained mean gate $\overline{|\alpha_\ell|}$, which falls monotonically with connectivity, mirroring the synthetic readout under real, heterogeneous graphs; this is consistent with an adaptive-graph model adjusting how much it relies on the graph to dataset-level graph quality through gradient descent alone. We do not ablate a gated substrate here (reasons are discussed in Appendix I).

6 Conclusion

Graph-based spatio-temporal forecasters share the substrate paradigm, a structural commitment that lets noise from uninformative adjacency components compound multiplicatively through depth. We proposed *spatial-as-refinement* as a structural alternative: graph operators contribute only through α-gated residual updates to a complete forecast, yielding a bound that is exactly zero when gates are closed and controllable, layer by layer, as they open. Instantiated as **GraFT**, this principle is validated by all three of the theory's predictions under both synthetic and real conditions, and a single instantiation establishes state-of-the-art performance against the strongest published baselines across fourteen real-world benchmarks spanning four domains. The choice was never graph or no graph; it was substrate or refinement. Once the graph's contribution is gated instead of being interleaved, misspecified adjacency no longer forces a multiplicative price.

AI USE STATEMENT

LLMs (GPT, Claude) supported LaTeX formatting, prose editing, and debugging of experimental scripts. All theory, algorithm design, and experimental protocols come from the authors. LLM-generated text was reviewed and revised as needed, and at least two authors verified each piece of LLM-generated code before inclusion. The authors are fully responsible for the paper's content.

ETHICS STATEMENT

This work proposes a general architectural principle for spatio-temporal forecasting and evaluates it on publicly released benchmarks, all aggregated at the level of sensors, stations, or zones, with no human subjects and no individual-level data. The societal impact is indirect and application-dependent. Improved forecasting could benefit domains such as traffic management, energy planning, and air-quality monitoring, but forecasts can also be over-trusted in consequential decisions, particularly on fragmented or sparsely instrumented networks where graph-based models are least reliable. We view GraFT as a methodological contribution whose real-world impact depends on how carefully downstream systems validate and oversee its predictions.

REPRODUCIBILITY STATEMENT

Real-data experiments are reproducible from the anonymous repository at this link, which contains the three GraFT architectures (MLP, PatchTST, TSMixer), the training and evaluation code, and one launch script per backbone and dataset that fixes the hyperparameters and random seed. Section 3.4 specifies the architecture, Section 5.1 and Appendix E describe the datasets and chronological splits, and Appendix F gives the training protocol. Appendix B contains complete proofs of all formal results, and Appendix C details the synthetic study.

REFERENCES

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W. Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 161 of *Proceedings of Machine Learning Research*, pp. 1352–1361, 2021.

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 (NeurIPS)*, volume 33, pp. 17804–17815, 2020.

Lars Ødegaard Bentsen, Narada Dilp Warakagoda, Roy Stenbro, and Paal Engelstad. A unified graph formulation for spatio-temporal wind forecasting. *Energies*, 16(20):7179, 2023.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

Ling Chen, Jiahui Xu, Binqing Wu, Yuntao Qian, Zhenhong Du, Yansheng Li, and Yongjun Zhang. Group-aware graph neural network for nationwide city air quality forecasting. *ACM Transactions on Knowledge Discovery from Data*, 18(3):1–20, 2023a.

Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O. Arik, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecasting. *Transactions on Machine Learning Research (TMLR)*, 2023b.

Yuchen Fang, Yanjun Qin, Haiyong Luo, Fang Zhao, Bingbing Xu, Chenxing Wang, and Liang Zeng. When spatio-temporal meet wavelets: Disentangled traffic forecasting via efficient spectral graph attention networks. In *Proceedings of the 39th IEEE International Conference on Data Engineering (ICDE)*, 2023.

Yuchen Fang, Yuxuan Liang, Bo Hui, Zezhi Shao, Liwei Deng, Xu Liu, Xinke Jiang, and Kai Zheng. Efficient large-scale traffic forecasting with transformers: A spatial data management perspective.

In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2025.

Jindong Han, Weijia Zhang, Hao Liu, Tao Tao, Naiqiang Tan, and Hui Xiong. Bigst: Linear complexity spatio-temporal graph neural network for traffic forecasting on large-scale road networks. *Proceedings of the VLDB Endowment*, 17(5):1081–1090, 2024.

Daejin Kim, Youngin Cho, Dongmin Kim, Cheonbok Park, and Jaegul Choo. Residual correction in real-time traffic forecasting. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 962–971, 2022.

Weiyang Kong, Kaiqi Wu, Sen Zhang, and Yubao Liu. Graphsparsenet: A novel method for large scale traffic flow prediction. *Proceedings of the VLDB Endowment*, 18, 2025.

Fuxian Li, Jie Feng, Huan Yan, Guangyin Jin, Depeng Jin, and Yong Li. Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 17(1):1–21, 2023.

Han Li, Haohao Qu, Xiaojun Tan, Linlin You, Rui Zhu, and Wenqi Fan. Urbanev: An open benchmark dataset for urban electric vehicle charging demand prediction. *Scientific Data*, 12:523, 2025.

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

Yuanzhe Li, Ao Zhou, Xiao Ma, and Shangguang Wang. Profit-aware edge server placement. *IEEE Internet of Things Journal*, 9(1):55–67, 2022.

Hangchen Liu, Zheng Dong, Renhe Jiang, Jiewen Deng, Jinliang Deng, Quanjun Chen, and Xuan Song. Spatio-temporal adaptive embedding makes vanilla transformer sota for traffic forecasting. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 4125–4129, 2023a.

Xu Liu, Yutong Xia, Yuxuan Liang, Junfeng Hu, Yiwei Wang, Lei Bai, Chao Huang, Zhenguang Liu, Bryan Hooi, and Roger Zimmermann. Largest: A benchmark dataset for large-scale traffic forecasting. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023b.

Jiecheng Lu, Xu Han, Yan Sun, and Shihao Yang. CATS: Enhancing multivariate time series forecasting by constructing auxiliary time series as exogenous variables. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Jiecheng Lu, Yan Sun, and Shihao Yang. In-context time series predictor. In *International Conference on Learning Representations (ICLR)*, 2025.

Jiaming Ma, Binwu Wang, Guanjun Wang, Kuo Yang, Zhengyang Zhou, Pengkun Wang, Xu Wang, and Yang Wang. Less but more: Linear adaptive graph learning empowering spatiotemporal forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2020.

Chao Shang, Jie Chen, and Jinbo Bi. Discrete graph structure learning for forecasting multiple time series. In *International Conference on Learning Representations (ICLR)*, 2021.

Zezhi Shao, Zhao Zhang, Fei Wang, Wei Wei, and Yongjun Xu. Spatial-temporal identity: A simple yet effective baseline for multivariate time series forecasting. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 4454–4458, 2022a.

Zezhi Shao, Zhao Zhang, Fei Wang, and Yongjun Xu. Pre-training enhanced spatial-temporal graph neural network for multivariate time series forecasting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1567–1577, 2022b.

Zezhi Shao, Zhao Zhang, Wei Wei, Fei Wang, Yongjun Xu, Xin Cao, and Christian S. Jensen. Decoupled dynamic spatial-temporal graph neural network for traffic forecasting. *Proceedings of the VLDB Endowment*, 15(11):2733–2746, 2022c.

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*, volume 34, pp. 914–921, 2020.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 32–42, 2021.

Artur Trindade. Electricity load diagrams 2011–2014, 2015.

Shuo Wang, Yanran Li, Jiang Zhang, Qingye Meng, Lingwei Meng, and Fei Gao. PM2.5-GNN: A domain knowledge enhanced graph neural network for PM2.5 forecasting. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, 2020. doi: 10.1145/3397536.3422208.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations (ICLR)*, 2023.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1907–1913, 2019.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pp. 753–763, 2020.

Chin-Chia Michael Yeh, Yujie Fan, Xin Dai, Uday Singh Saini, Vivian Lai, Prince Osei Aboagye, Junpeng Wang, Huiyuan Chen, Yan Zheng, Zhongfang Zhuang, Liang Wang, and Wei Zhang. Rpmixer: Shaking up time series forecasting with random projections for large spatial-temporal data. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 3919–3930, 2024.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3634–3640, 2018.

A Extended Related Work

Section 2 summarizes the related work in a single paragraph; this appendix gives the full discussion, organized by theme, that the summary compresses.

Substrate spatio-temporal GNNs. The dominant family of spatio-temporal forecasters follows the substrate template of Section 3.2. STGCN and DCRNN couple GCN-style spatial aggregation with temporal convolutions or recurrences; GraphWaveNet, AGCRN, and MTGNN introduce learnable adaptive adjacencies; DGCRN constructs the graph dynamically from inputs; D^{2}STGNN decouples diffusion and inherent signals; STAEformer adds spatio-temporal attention; and MAGE introduces a mixture of kernel-trick experts. All differ in what the graph operator computes but preserve where it acts: at every layer, interleaved with each temporal block. Several of these models (GraphWaveNet, MTGNN, D^{2}STGNN, STAEformer) already wrap their graph operator in a residual or skip connection. This alone does not give the optimizer a silencing mechanism: the residual coefficient is fixed, or, when learnable, entangled with the same weights that carry the informative signal, so suppressing a noisy layer still requires shrinking $\mathcal{G}_\ell^\star$ itself, the "no silencing mechanism" fact of Section 3.2(ii). What GraFT adds is not the residual topology but a dedicated, zero-initialized scalar α_ℓ, decoupled from $\mathcal{G}_\ell$'s own weights, that the optimizer can move independently. GraFT departs along this second axis, placing the graph operator after a complete temporal forecast, as refinement instead of substrate.

Efficient adaptive graph learning. A parallel line targets the N^2 cost of dense adjacencies: BigST (Han et al., 2024) and GSNet (Kong et al., 2025) exploit sparsity, while MAGE's kernel trick avoids materializing the $N \times N$ matrix by factoring the adjacency through low-dimensional node embeddings. GraFT's operator normalizes both embeddings across modes; MAGE's kernel-trick factorization normalizes the second across nodes instead, so the two are reached differently: MAGE treats its row-stochastic form as a low-rank approximation to a full-rank adaptive graph, recovered with a mixture of experts, whereas we derive ours bottom-up from a single requirement, that every sensor route through a shared, capacity-conserving bottleneck (Lemma 1), so ours is mathematically derived, not approximated. We inherit neither MAGE's approximation framing nor its expert mixing; holding the underlying computation's shape fixed is what lets the synthetic study (Section 4) isolate the two differences that remain: placement and α-gating.

Graph-free spatio-temporal forecasters. A more recent line argues strong per-node temporal models can match or exceed graph-based methods: STID (Shao et al., 2022a) with a shallow MLP and identity embeddings, RPMixer (Yeh et al., 2024) with random projection mixing, and PatchSTG with geography-based patching in place of a learned graph. These methods expose the tension GraFT targets (leading performance on fragmented graphs without graph operators) but resolve it by discarding relational signal where the graph is informative. GraFT keeps that signal available and lets the data decide whether to use it.

Two-stage forecasting architectures. A separate line splits forecasting into two explicit stages instead of interleaving spatial and temporal computation at every layer. STEP (Shao et al., 2022b) pretrains a masked-autoencoding temporal module and freezes it before a downstream GNN consumes the embeddings, so its first stage is not a forecaster and the graph stays a substrate, and ResCAL (Kim et al., 2022) is a post-hoc calibrator, training a residual estimator (itself a graph-convolutional substrate) on a separately trained forecaster's past errors; GraFT differs from both by training predictor and refinement jointly, with refinement acting on latents and no error feedback. GTS (Shang et al., 2021) jointly trains structure learning and forecasting, but its sampled graph is the adjacency of a DCRNN that mixes across sensors at every step, whereas in GraFT the forecast exists before the graph acts. N-BEATS (Oreshkin et al., 2020) performs graph-free stacking of univariate temporal blocks; GraFT's graph blocks instead run across sensors, correcting a completed forecast.

Gated residual methods. A different line uses learned gates to ease the optimization of deep residual networks. ReZero (Bachlechner et al., 2021) and LayerScale (Touvron et al., 2021) gate residual branches uniformly, as optimization aids for depth: ReZero scales every branch by a zero-initialized scalar, so with all gates closed the network collapses to the identity, and LayerScale scales each channel of every branch by a small learned diagonal. GraFT keeps the zero-initialized scalar but gates selectively: only the cross-sensor branch, so closing every gate leaves a complete temporal forecaster (Proposition 2). STUGN (Bentsen et al., 2023) gates graph blocks that carry the whole spatio-temporal computation atop a persistence baseline, so a fully closed STUGN leaves only per-

sistence, whereas in GraFT the forecast is complete before the graph acts, so its gates isolate the cross-sensor channel, bound the contamination (Proposition 4), and read out how much the graph helps.

Temporal-only forecasting backbones. GraFT inherits its per-sensor predictor from time-series forecasters showing that a sufficiently expressive shared temporal backbone can compete without architectural specialization; PatchTST's patch-based transformer and TSMixer's all-MLP mixing serve directly as backbones in Section 5. GraFT uses such a backbone as the foundation ϕ_θ and not the entire model, adding gated spatial refinement on top. Non-graph multivariate forecasters can also inject cross-series context: CATS (Lu et al., 2024) constructs auxiliary series from the input as exogenous variables for a simple MLP predictor, capturing inter-series relationships without an adjacency. There, the cross-series signal is built into the predictor's input, whereas GraFT adds it as a gated correction to a complete per-sensor forecast.

B Full Statements and Proofs

B.1 Proof of Proposition 1 (Substrate multiplicative noise bound)

We restate the bound without the $T = \max_k \|\mathcal{T}_k\|_{\mathrm{op}}$ relaxation, retaining the interleaved temporal factors.

Proposition 6 (Substrate bound, non-asymptotic form)**.** *Let $\mathcal{T}_\ell, \mathcal{G}_\ell^\star, W_{\mathrm{out}}$ be bounded linear operators and $\mathcal{G}_\ell = \mathcal{G}_\ell^\star + \mathcal{N}_\ell$ with $\|\mathcal{N}_\ell\|_{\mathrm{op}} \le \epsilon$. Write the noise-free output $\hat{y}^\star = W_{\mathrm{out}} \prod_{\ell=L}^{1} (\mathcal{T}_\ell \mathcal{G}_\ell^\star)\, h^{(0)}$. Then*

$$\big\|\hat{y} - \hat{y}^\star\big\| \;\le\; \|W_{\mathrm{out}}\|_{\mathrm{op}}\, \|h^{(0)}\| \sum_{\ell=1}^{L} \Big(\prod_{k>\ell} \|\mathcal{T}_k\|_{\mathrm{op}} \|\mathcal{G}_k^\star\|_{\mathrm{op}}\Big)\, \|\mathcal{T}_\ell\|_{\mathrm{op}}\, \epsilon\, \Big(\prod_{k<\ell} \|\mathcal{T}_k\|_{\mathrm{op}} \|\mathcal{G}_k^\star\|_{\mathrm{op}}\Big) + O(\epsilon^2). \tag{16}$$

Bounding every $\|\mathcal{T}_k\|_{\mathrm{op}} \le T$ and $\|\mathcal{G}_k^\star\|_{\mathrm{op}} \le G$ collapses equation 16 to the main-text form equation 4, which is $O(L\epsilon T^L G^{L-1})$. The $O(\epsilon^2)$ remainder is sign-definite in the same direction: it is a sum of products containing two or more factors of $\mathcal{N}_k$, each bounded by ϵ and multiplied by additional $\mathcal{G}_k^\star$ norms, so retaining it can only increase the right-hand side.

Proof. Write the noisy and clean forward maps as the operator products $P = \prod_{\ell=L}^{1} (\mathcal{T}_\ell \mathcal{G}_\ell)$ and $P^\star = \prod_{\ell=L}^{1} (\mathcal{T}_\ell \mathcal{G}_\ell^\star)$, so $\hat{y} - \hat{y}^\star = W_{\mathrm{out}}(P - P^\star)h^{(0)}$. Substituting $\mathcal{G}_\ell = \mathcal{G}_\ell^\star + \mathcal{N}_\ell$ and expanding the product, the terms linear in the $\mathcal{N}_\ell$ are

$$P - P^\star = \sum_{\ell=1}^{L} \Big(\prod_{k>\ell} \mathcal{T}_k \mathcal{G}_k^\star\Big)\, \mathcal{T}_\ell \mathcal{N}_\ell \,\Big(\prod_{k<\ell} \mathcal{T}_k \mathcal{G}_k^\star\Big) \;+\; R, \tag{17}$$

where R collects all terms containing two or more factors $\mathcal{N}_k$. Applying submultiplicativity of $\|\cdot\|_{\mathrm{op}}$, the bound $\|\mathcal{N}_\ell\|_{\mathrm{op}} \le \epsilon$, and $\|W_{\mathrm{out}}(\cdot)h^{(0)}\| \le \|W_{\mathrm{out}}\|_{\mathrm{op}} \|h^{(0)}\| \, \|\cdot\|_{\mathrm{op}}$ to the linear sum yields the displayed right-hand side of equation 16. For the remainder, every term in R is a product of the same bounded operators with at least two factors $\mathcal{N}_k$, hence $\|R\|_{\mathrm{op}} = O(\epsilon^2)$ with nonnegative contribution to the norm bound; this gives the final claim and the sign statement. □

Remark 2. *The structural reading of equation 16 is unchanged from the main text: once the temporal operators are bounded by T, each summand still carries a product of $L-1$ graph-operator norms $\prod_{k\neq\ell} \|\mathcal{G}_k^\star\|_{\mathrm{op}}$, so noise injected at depth ℓ is amplified by the operator norms of every other layer. No single scalar multiplies $\mathcal{N}_\ell$ in isolation, which is the formal content of the "no silencing mechanism" fact: suppressing $\mathcal{N}_\ell$ requires shrinking $\mathcal{G}_\ell = \mathcal{G}_\ell^\star + \mathcal{N}_\ell$ as a whole, removing the informative $\mathcal{G}_\ell^\star$ with it.*

B.2 Proof of Proposition 4 (Refinement gated noise bound)

Proposition 7 (Refinement bound, restated for proof)**.** *Consider the LayerNorm-free refinement recursion $y^{(\ell+1)} = y^{(\ell)} + \alpha_\ell\, \mathcal{G}_\ell(y^{(\ell)})$ with $\mathcal{G}_\ell = \mathcal{G}_\ell^\star + \mathcal{N}_\ell$, $\|\mathcal{N}_\ell\|_{\mathrm{op}} \le \epsilon$, and let $y^{(\ell)\star}$ be the*

trajectory generated by the same recursion with every $\mathcal{G}_\ell$ replaced by $\mathcal{G}_\ell^\star$, started from the common $y^{(0)} = y^{(0)\star} = \phi_\theta(x)$. Then for all $\epsilon \geq 0$,

$$\big\|\hat{y} - \hat{y}^\star\big\| \;\leq\; \|W_{\text{out}}\|_{\text{op}}\, \epsilon \sum_{\ell=0}^{L-1} |\alpha_\ell|\, \|y^{(\ell)}\| \prod_{j=\ell+1}^{L-1} \big(1 + |\alpha_j|\, \|\mathcal{G}_j^\star\|_{\text{op}}\big). \tag{18}$$

Proof. Let $\delta_\ell = \|y^{(\ell)} - y^{(\ell)\star}\|$, so $\delta_0 = 0$. Subtracting the two recursions,

$$y^{(\ell+1)} - y^{(\ell+1)\star} = \big(y^{(\ell)} - y^{(\ell)\star}\big) + \alpha_\ell \Big(\mathcal{G}_\ell(y^{(\ell)}) - \mathcal{G}_\ell^\star(y^{(\ell)\star})\Big). \tag{19}$$

Decompose the bracketed difference as

$$\mathcal{G}_\ell(y^{(\ell)}) - \mathcal{G}_\ell^\star(y^{(\ell)\star}) = \mathcal{G}_\ell^\star\big(y^{(\ell)} - y^{(\ell)\star}\big) + \mathcal{N}_\ell\big(y^{(\ell)}\big), \tag{20}$$

which holds since $\mathcal{G}_\ell = \mathcal{G}_\ell^\star + \mathcal{N}_\ell$ is linear. Taking norms and using the triangle inequality, submultiplicativity, and $\|\mathcal{N}_\ell\|_{\text{op}} \leq \epsilon$,

$$\delta_{\ell+1} \;\leq\; \delta_\ell\big(1 + |\alpha_\ell|\, \|\mathcal{G}_\ell^\star\|_{\text{op}}\big) + |\alpha_\ell|\, \epsilon\, \|y^{(\ell)}\|. \tag{21}$$

This is a linear recursion in δ_ℓ with multiplier $m_\ell = 1 + |\alpha_\ell|\, \|\mathcal{G}_\ell^\star\|_{\text{op}}$ and forcing term $f_\ell = |\alpha_\ell|\, \epsilon\, \|y^{(\ell)}\|$. Unrolling from $\delta_0 = 0$,

$$\delta_L \;\leq\; \sum_{\ell=0}^{L-1} f_\ell \prod_{j=\ell+1}^{L-1} m_j \;=\; \epsilon \sum_{\ell=0}^{L-1} |\alpha_\ell|\, \|y^{(\ell)}\| \prod_{j=\ell+1}^{L-1} \big(1 + |\alpha_j|\, \|\mathcal{G}_j^\star\|_{\text{op}}\big). \tag{22}$$

Applying $\|\hat{y} - \hat{y}^\star\| = \|W_{\text{out}}(y^{(L)} - y^{(L)\star})\| \leq \|W_{\text{out}}\|_{\text{op}}\, \delta_L$ gives equation 18. The persistence skip $x_{t,\cdot,\text{val}}$ is identical in $\hat{y}$ and $\hat{y}^\star$ and cancels in the difference, so it does not appear. No step discards ϵ^2 terms: equation 21 is an exact consequence of the triangle inequality, so equation 18 holds for all $\epsilon \geq 0$, not merely to first order, in contrast to the first-order substrate bound of Proposition 6. □

Remark 3 (Corollaries)**.** *Setting $\alpha_\ell = 0$ for all ℓ makes every forcing term f_ℓ vanish, so $\delta_L = 0$ and contamination is exactly zero (corollary (a)). Each $\mathcal{G}_j^\star$ satisfies $\|\mathcal{G}_j^\star\|_{\text{op}} \leq M_j$ for a per-layer, directly computable constant M_j by Lemma 1, and if $|\alpha_\ell| \leq \alpha_{\max} \ll 1/M_j$ (the near-initialization regime, not the trained operating point; Section 5.3 reports trained magnitudes of* 0.28–1.18*) then $\prod_{j>\ell}(1 + |\alpha_j| M_j) = 1 + O(\alpha_{\max})$, giving the small-gate form $\|\hat{y} - \hat{y}^\star\| \leq \|W_{\text{out}}\|_{\text{op}}\, \epsilon \sum_\ell |\alpha_\ell|\, \|y^{(\ell)}\| + O(\alpha_{\max}^2)$ (corollary (b)). At any gate magnitude the general bound equation 18 already applies, with every product factor an independently trainable scalar, not a fixed, entangled quantity as in the substrate case.*

B.3 Proof of Lemma 1 (Bounded mixing)

Sensor i's row $p_{i,:}^{(\ell)}$ is its decode-side distribution over modes and sensor j's row $q_{j,:}^{(\ell)}$ is its encode-side distribution over the same modes; both lie in $\Delta^{d_G - 1}$, so every sensor, on both sides, spends exactly one unit of routing mass split across the d_G modes. Equivalently, $\mathcal{G}_\ell$ is a width-d_G linear bottleneck: $\text{softmax}(E_2^{(\ell)})^\top y$ encodes the N sensor signals into d_G spatial-mode summaries, and $\text{softmax}(E_1^{(\ell)})$ decodes a per-sensor correction from them.

Proof. By equation 11, $|[\mathcal{G}_\ell(y)]_i| = \Big|\sum_j w_{ij}^{(\ell)} y_j\Big| \leq \Big(\sum_j w_{ij}^{(\ell)}\Big)\|y\|_\infty$ for every i, using $w_{ij}^{(\ell)} \geq 0$. Since $\sum_j w_{ij}^{(\ell)} = \sum_k p_{ik}^{(\ell)} \sum_j q_{jk}^{(\ell)} = \sum_k p_{ik}^{(\ell)}\, c_k^{(\ell)}$ is a convex combination of $\{c_k^{(\ell)}\}_k$ (since $p_{i,:}^{(\ell)} \in \Delta^{d_G - 1}$), it is at most $\max_k c_k^{(\ell)} = M_\ell$ for every i; hence $\|\mathcal{G}_\ell(y)\|_\infty \leq M_\ell \|y\|_\infty$, and the induced ℓ_∞ operator norm (the maximum absolute row sum) satisfies $\|\mathcal{G}_\ell\|_{\text{op}} \leq M_\ell$. Because M_ℓ is one of d_G non-negative numbers summing to N (the conservation identity of Section 3.4), $M_\ell \leq N$. M_ℓ costs nothing beyond what Proposition 5 already accounts for: it is exactly the column sum of $\text{softmax}(E_2^{(\ell)})$, a byproduct of the encode step in equation 13. □

C Synthetic Construction and Full Diagnostic Results

This appendix gives the full construction and results for the synthetic verification compressed in Section 4. The experiments are diagnostic: they isolate the structural consequence of *where* the graph operator is placed, holding the operator and predictor fixed, and test the falsifiable predictions P1–P3 together with the noise-propagation mechanism of Propositions 1 and 4.

C.1 Data Generating Process

We generate $N = 200$ sensors partitioned into $K = 4$ equal-size communities of 50 sensors each. Each community k carries a smooth, forecastable shared latent $s_k(t)$, formed from a small fixed set of sinusoids plus a mild AR(1) term. Each sensor i in community $k(i)$ observes this latent only through a sensor-specific near-Nyquist carrier:

$$x_i(t) = \rho\,\big(1 + c_i(t)\big)\, s_{k(i)}(t) + (1-\rho)\,\eta_i(t), \qquad c_i(t) = \cos(\omega_i t + \varphi_i), \tag{23}$$

with η_i i.i.d. white noise and per-sensor parameters $\omega_i \in [\pi/2, \pi]$, $\varphi_i \in [0, 2\pi)$ drawn once per sensor and held fixed.

The carrier c_i is zero-mean in time and, with phases drawn independently across the community, approximately zero-mean across sensors within a community. Two consequences follow, and together they make spatial mixing genuinely irreplaceable rather than a denoising convenience:

- **Each sensor is near-white on its own.** The near-Nyquist modulation scrambles the smooth latent so that x_i has low temporal autocorrelation; its own past carries little information about its future. A per-sensor temporal predictor therefore cannot recover s_k from a single sensor.
- **Community averaging recovers the latent.** Because the carriers cancel in expectation across the community,

$$\frac{1}{|k|}\sum_{i \in k} x_i(t) \;\approx\; \rho\, s_k(t), \tag{24}$$

  so the forecastable signal is accessible only by mixing across sensors in the same community. No amount of per-sensor temporal modeling can substitute for this aggregation.

The scalar $\rho \in [0, 1]$ controls graph informativeness. As $\rho \to 1$ the recoverable latent dominates and spatial mixing is maximally useful; as $\rho \to 0$ only the white-noise term η_i remains and nothing is forecastable by any means. Decreasing ρ thus maps onto a larger effective noise norm $\|\mathcal{N}_\ell\|_{\mathrm{op}}$ in the perturbed-adjacency model of Section 3.2: a graph that should aggregate within communities is increasingly contaminated by the un-forecastable component. We split each series 6:2:2 chronologically, sweep $\rho \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$, and set $T = H = 12$.

C.2 Models Compared

We compare two architectures at matched parameter budgets, sharing the same per-sensor predictor ϕ_θ and the same kernel-trick graph operator of Section 3.4:

- **Substrate.** The operator is applied at every layer, interleaved with the predictor, with no α-gating, matching the template of Section 3.2.
- **GraFT.** The full spatial-as-refinement architecture: the predictor produces a complete trajectory, refined by α-gated residual graph blocks.

Because the operator and predictor are identical, every behavioral difference between the two is attributable to placement and gating alone.

C.3 Behavioral Predictions (Full)

Table 3 reports the full sweep: test MAE for both architectures and the mean gate magnitude $\overline{|\alpha_\ell|}$ at every ρ. The substrate MAE is high and nearly flat (P1); GraFT improves monotonically with ρ and never exceeds the substrate (P2); and $\overline{|\alpha_\ell|}$ falls as the graph fragments (P3), collapsing toward zero at $\rho = 0.1$.

Table 3: Full synthetic sweep. Test MAE for Substrate and GraFT, and GraFT's mean gate magnitude $\overline{|\alpha_\ell|}$, across the fragmentation parameter ρ (larger ρ = more informative graph). Both models share the same predictor and operator; every difference is attributable to placement and gating.

ρ	0.10	0.25	0.50	0.75	0.90
Substrate MAE	1.130	1.127	1.104	1.069	1.051
GraFT MAE	0.800	0.790	0.705	0.571	0.508
GraFT $\overline{\lvert\alpha_\ell\rvert}$	0.002	0.184	0.201	0.212	0.233

C.4 Mechanistic Verification (Full)

The behavioral results confirm the predicted consequences: substrate MAE divergence, refinement graceful degradation, and gate closing. They do not confirm the mechanism that noise propagates multiplicatively in the substrate and through a gated channel in the refinement. To test the mechanism directly, we inject a perturbation $\mathcal{N}_\ell$ of known norm $\|\mathcal{N}_\ell\|_{\text{op}} = \epsilon$ and measure the resulting output change $\|\hat{y} - \hat{y}^\star\|$.

Probe path. We evaluate the perturbation on a probe path that realizes exactly the linear operator products that Propositions 1 and 4 bound: the temporal operators are set to the identity, and LayerNorm and the FFN are removed, so that the measured contamination reflects only the graph operators and, for GraFT, the gates. For the substrate we sweep stack depth $L \in \{2, 3, 4\}$; for GraFT we inject into each refinement layer of a trained model in turn.

Substrate: multiplicative compounding. Contamination grows linearly in ϵ, confirming the first-order theory, and its level explodes with depth: at $\epsilon = 1.6$ the output contamination rises from ≈ 1.1 at $L = 2$ to ≈ 11.8 at $L = 3$ to ≈ 169.8 at $L = 4$, roughly an order of magnitude per added layer, which matches the $\prod_k \|\mathcal{G}_k^\star\|_{\text{op}}$ scaling predicted by Proposition 1 (here every $\|\mathcal{G}_k^\star\|_{\text{op}} \approx 12.5$ in the unnormalized probe). Injecting the same ϵ at a deeper substrate produces strictly larger contamination, the direct empirical signature of multiplicative compounding.

GraFT: gate-controlled structure. Each refinement layer contaminates the output linearly in ϵ, with slope ordered exactly by its trained gate: layers with $|\alpha_\ell| \approx 0.24, 0.20, 0.15$ produce progressively smaller contamination (at $\epsilon = 1.6$, roughly 1.56, 0.73, and 0.25 respectively), with no amplification through the rest of the stack. This is precisely the gated bound of Proposition 4: a layer whose gate is small is effectively silenced regardless of ϵ, while an open gate's contamination increases with its own magnitude.

The contrast between the two regimes is the load-bearing empirical statement of the synthetic study: substrate noise reaches the output through a multiplicative channel the architecture cannot suppress, while refinement noise reaches it through a gated channel the optimizer can, and does, close on a per-layer basis. Section 5 tests whether the same mechanism persists under heterogeneous real-world conditions.

D Network Connectivity

We characterize each dataset's connectivity, using the raw physical adjacency for this analysis only (the model never uses it), as the cross-community edge fraction (CCEF): the fraction of total edge weight that crosses community boundaries, with communities detected via weighted Louvain modularity optimization (Blondel et al., 2008). Given the symmetrized adjacency $A_{\text{sym}} = \max(A, A^\top)$ with zero diagonal, we build a weighted undirected graph G, partition its nodes into communities $\{C_k\}$ by maximizing Louvain modularity, and compute

$$\text{CCEF} = \frac{\sum_{(i,j)\in G:\, c(i)\neq c(j)} w_{ij}}{\sum_{(i,j)\in G} w_{ij}}, \tag{25}$$

where $c(i)$ denotes the community containing node i and w_{ij} the weight of edge (i, j). A high CCEF means most edge weight connects different communities: the graph is well-mixed and physically connected. A low CCEF means edge weight is concentrated within isolated communities, the signature of a fragmented network, such as GBA's disjoint highway corridors. Unlike a simple

edge-density count, CCEF remains well-defined for disconnected graphs and specifically penalizes corridor-level isolation and not sparsity alone. Louvain is stochastic, so we average CCEF over multiple random seeds (10 for graphs with $N \leq 3{,}000$ nodes, 3 for larger graphs) and treat a benchmark's connectivity as stable when the seed-to-seed standard deviation is below 0.005; this held for all benchmarks tested. The values reported as "Connectivity" in Table 2 are the resulting CCEF.

E DATASETS

Table 4: **Dataset statistics.** M: Million (10^6). B: Billion (10^9). Data points is the product of node count and total timesteps.

Domain	Dataset	#Nodes	#Edges	Data Points	Res.	Time Span
Traffic	PEMS03	358	546	9.38M	5 min	2018
	PEMS04	307	338	5.22M	5 min	2018
	PEMS07	883	865	24.92M	5 min	2017
	PEMS08	170	276	3.04M	5 min	2016
	PEMS-Bay	325	2,369	16.94M	5 min	2017
	LargeST-SD	716	17,319	0.38B	5 min	2017–2021
	LargeST-GBA	2,352	61,246	1.24B	5 min	2017–2021
	LargeST-GLA	3,834	98,703	2.02B	5 min	2017–2021
	LargeST-CA	8,600	201,363	4.52B	5 min	2017–2021
Energy	Electricity	321	101,323	8.44M	1 hr	2012–2014
	UrbanEV	1,682	1,989,840	14.73M	1 hr	2022–2023
Meteorology	KnowAir	184	3,796	2.15M	3 hr	2015–2018
	China City Air Quality	209	4,321	4.26M	1 hr	2017–2019
Mobility	Shanghai Mobile	3,042	9,090,300	13.36M	1 hr	2014

Table 4 summarizes node count, edge count, sampling resolution, time span, and total data points (nodes $\times$ timesteps) for all fourteen benchmarks, grouped by domain. All datasets are split chronologically 6:2:2 into train, validation, and test, with a 12-step input/horizon for the traffic benchmarks and a 24-step input/horizon for all others.

Traffic. PEMS03, PEMS04, PEMS07, PEMS08, and PEMS-Bay are highway sensor networks released by Caltrans' Performance Measurement System (PeMS), each sampled every 5 minutes. LargeST-SD, GBA, GLA, and CA (Liu et al., 2023b) are four larger PeMS-derived splits at increasing spatial scale, from San Diego (SD) up to statewide California (CA), ranging from 716 to 8,600 sensors.

Energy. Electricity (Trindade, 2015) is a standard multivariate forecasting benchmark recording hourly consumption for 321 clients. UrbanEV (Li et al., 2025) is a real-world electric-vehicle charging dataset from Shenzhen, aggregated into a spatial graph of 1,682 charging zones at hourly resolution.

Meteorology. KnowAir (Wang et al., 2020) records $PM_{2.5}$ and accompanying meteorological attributes at 184 monitoring stations across China at 3-hour resolution. China City Air Quality (Chen et al., 2023a) covers a larger 209-city network at hourly resolution.

Mobility. Shanghai Mobile (Li et al., 2022) records mobile internet access through base stations in Shanghai, aggregated hourly.

F IMPLEMENTATION AND TRAINING

All models are trained with AdamW, with an exponential moving average of the weights (decay 0.999) used for evaluation. Learning rate follows a linear warmup for 5 epochs (from 10^{-4} to a peak of 0.001–0.002, tuned per dataset) followed by linear decay to 5×10^{-5}; weight decay is 0.0001. Batch size is fixed at 16 across every dataset and backbone, and models train for 30–60 epochs depending on dataset with early stopping after 7 epochs without validation improvement. Gradients are clipped at a global norm of 5 and dropout is 0 for most configurations. The predictor's

output dimension is $d = 128$ for the MLP backbone and $d = 64$ for PatchTST and TSMixer; the linear adaptive graph operator uses $d_G \in \{64, 96, 128\}$ spatial modes and gate α_ℓ initialized to zero (Section 3.4), with refinement depth $L \in \{1, 2, 3\}$ tuned per dataset. All reported results are the mean over 5 seeds. Training runs on a single NVIDIA RTX A6000 GPU. Exact per-dataset hyperparameters (learning rate peaks, refinement depth L) and the tuning grids searched are provided in the released training scripts and configuration files.

PatchTST and TSMixer instantiations. Both backbones are implemented following the patch-based and mixer-based designs made widely available through the Time-Series-Library (Wu et al., 2023), adapted here into drop-in replacements for the MLP predictor ϕ_θ: same input (raw signal concatenated with time-of-day, day-of-week, and node embeddings) and same output shape, so the downstream refinement stack, output head, and persistence skip are unchanged. Only ϕ_θ's internals differ, consistent with the backbone-agnostic claim of Section 3.4.

The **PatchTST** backbone patchifies the input window into overlapping length-2 patches (stride 1), linearly embeds each patch to d_{model}, adds a learned positional embedding, and processes the patch sequence with a 2-layer pre-LayerNorm Transformer encoder (GELU activation, feed-forward width $2d_{\text{model}}$, attention heads set to the largest divisor of d_{model} in $\{8, 4, 2, 1\}$); a linear head flattens all patch representations to the horizon output. Because PatchTST's flatten head scales with hidden width, its hidden dimension is reduced relative to the MLP and TSMixer backbones to keep parameter budgets comparable across the three instantiations.

The **TSMixer** backbone stacks 2 mixer blocks, each alternating a time-mixing MLP (LayerNorm, then a linear-GELU sub-layer applied along the time axis, added back as a residual) and a feature-mixing MLP (LayerNorm, then a two-layer GELU sub-layer along the channel axis, added back as a residual), followed by a temporal projection ($T \to H$) and a channel projection to the output width.

G Full RMSE and MAPE Results

Table 5 reports MAE, RMSE, and MAPE for every model and benchmark, stacked by domain group. #Wins and AvgRank are computed jointly across all fourteen benchmarks and all three metrics; instead of repeating this identical summary in every block, we report it once, in the bottom block.

Table 5: **Full MAE/RMSE/MAPE across all fourteen benchmarks, by domain group.** **Best** (red) and second best (blue) per column, matching the convention of Table 1. #Wins and AvgRank are computed jointly across all three blocks below and reported once, in the bottom block. Values are the mean over 5 seeds; "/" = PatchSTG only, not reported due to missing latitude/longitude metadata required by its patching mechanism; "–" = out-of-memory even at the minimum batch size.

LargeST

Model	SD			GLA			GBA			CA		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
GraFT (PatchTST)	16.64	29.06	10.91	19.23	32.32	**11.57**	**19.09**	**32.29**	**14.50**	**17.78**	**29.89**	13.12
GraFT (MLP)	16.72	28.26	**10.76**	19.37	**31.93**	11.61	19.51	32.63	15.12	17.87	30.28	**12.99**
GraFT (TSMixer)	17.02	29.41	11.39	20.12	33.20	12.37	20.35	33.64	15.78	18.74	31.18	13.95
MAGE	**16.35**	**28.06**	10.93	19.74	32.74	12.12	19.84	33.32	15.52	18.28	30.33	14.12
PatchSTG	16.98	28.84	11.48	**19.19**	32.84	11.70	19.67	33.36	15.18	17.99	30.66	13.16
STAEformer	19.02	31.78	12.65	–	–	–	21.30	34.56	17.63	–	–	–
GWNet	18.07	29.97	12.70	20.37	32.65	12.71	20.83	33.37	17.30	19.75	31.71	15.84
DGCRN	17.79	29.31	12.33	–	–	–	20.53	33.40	16.79	–	–	–
D^2STGNN	17.13	28.60	12.15	–	–	–	21.13	34.09	16.08	–	–	–
STWave	17.64	29.61	11.83	20.22	33.03	12.38	20.56	33.58	15.14	20.67	33.12	15.76
BigST	17.68	29.61	11.66	20.98	34.40	13.30	21.15	34.38	17.80	19.32	32.01	14.93
STID	18.03	30.85	12.18	20.40	33.90	12.97	20.65	34.29	16.92	19.04	31.86	14.69
AGCRN	18.39	33.63	13.78	20.26	34.86	12.39	20.69	34.30	16.05	–	–	–
MTGNN	18.21	30.99	12.36	21.75	35.35	14.88	21.48	34.91	17.17	19.91	32.63	15.11
GTS	18.34	30.34	12.09	–	–	–	22.06	35.33	18.30	–	–	–

PEMS

Model	PEMS03			PEMS04			PEMS07			PEMS08			PEMS-Bay		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
GraFT (PatchTST)	**14.34**	24.42	**14.94**	**18.14**	29.79	**12.40**	**18.90**	**32.51**	**7.91**	13.79	23.17	**8.96**	**1.54**	**3.49**	**3.51**
GraFT (MLP)	14.53	24.80	14.96	18.20	29.81	12.44	19.22	33.10	8.08	13.87	23.22	9.08	1.57	3.56	3.57
GraFT (TSMixer)	14.87	24.22	15.79	18.38	29.93	12.65	19.54	32.97	8.21	14.26	23.22	9.25	1.58	3.58	3.61
MAGE	14.66	**23.51**	15.82	18.25	30.33	12.72	19.63	32.83	8.28	13.93	**23.09**	9.28	1.59	3.52	3.57
PatchSTG	/	/	/	/	/	/	/	/	/	/	/	/	1.61	3.74	3.65
STAEformer	15.27	26.76	15.88	18.78	30.30	13.06	20.09	33.36	8.41	14.17	23.38	9.18	1.65	3.61	3.75
GWNet	14.76	25.35	15.38	18.81	30.29	13.06	19.92	32.84	8.62	14.20	23.13	9.53	1.61	3.61	3.63
DGCRN	15.09	26.02	16.05	18.68	30.21	13.04	20.24	33.08	8.71	14.34	23.53	9.48	1.64	3.67	3.66
D^2STGNN	14.84	25.41	15.17	18.61	30.13	12.82	20.33	33.23	8.73	14.36	23.46	9.32	1.62	3.69	3.68
STWave	14.89	26.89	15.15	18.69	30.50	12.67	20.11	33.47	8.40	**13.74**	23.45	8.99	1.65	3.70	3.74
BigST	15.30	25.77	16.54	18.42	29.96	12.92	20.31	33.57	8.57	14.19	23.26	9.29	1.65	3.58	3.77
STID	15.37	26.39	16.57	18.29	**29.74**	12.45	19.54	32.54	8.28	14.19	23.28	9.25	1.70	3.86	3.91
AGCRN	15.60	26.88	15.25	19.25	31.10	13.00	20.40	34.24	8.62	15.54	24.77	10.15	1.65	3.68	3.75
MTGNN	15.31	25.95	15.04	19.20	31.81	13.26	20.97	34.20	8.90	15.22	24.09	9.89	1.63	3.66	3.62
GTS	14.99	25.81	15.34	20.33	32.12	14.23	21.34	34.40	9.03	15.40	24.61	9.92	1.67	3.81	3.82

Mobility, Meteorology, and Energy — #Wins and AvgRank below are computed jointly over all three blocks above

Model	ShanghaiMobile			KnowAir			China Air Quality			UrbanEV			Electricity			**#Wins**	**AvgRank**
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE		
GraFT (PatchTST)	0.386	0.757	20.02	15.58	24.25	55.83	17.94	30.30	28.25	**3.81**	18.37	43.24	**172.4**	1571.0	**10.72**	**20**	**2.10**
GraFT (MLP)	**0.383**	0.755	19.83	15.46	23.91	55.48	**17.88**	**30.11**	28.22	3.97	19.20	49.55	177.8	**1488.4**	11.49	7	2.48
GraFT (TSMixer)	0.386	0.758	19.92	**15.20**	**23.34**	55.44	17.97	30.48	28.11	3.94	19.22	45.96	182.2	1559.0	12.68	2	4.39
MAGE	0.395	0.763	19.86	16.65	25.03	61.76	18.86	31.68	29.55	3.88	**17.72**	49.14	181.0	1634.6	12.78	5	4.64
PatchSTG	0.389	**0.753**	19.98	16.54	25.24	63.29	19.06	32.25	29.43	4.74	20.81	86.43	/	/	/	2	6.30
STAEformer	–	–	–	16.49	25.56	61.37	19.01	31.57	30.34	4.61	20.37	76.25	200.5	1650.0	13.75	0	9.64
GWNet	0.398	0.790	**19.03**	15.80	24.14	61.01	18.74	31.72	29.11	4.50	19.78	56.91	200.3	1820.0	13.48	1	7.38
DGCRN	–	–	–	21.11	30.62	65.89	21.87	35.18	35.05	7.09	26.92	110.45	250.3	2353.0	18.14	0	10.82
D^2STGNN	–	–	–	15.96	25.16	58.19	18.82	32.29	**26.30**	4.31	18.42	56.80	233.2	2177.6	16.00	1	7.70
STWave	–	–	–	16.46	26.37	**54.79**	20.26	33.95	32.07	4.30	19.34	52.02	190.1	1736.3	13.63	2	7.77
BigST	0.394	0.761	19.57	17.02	24.83	69.46	18.67	31.02	29.37	4.78	19.42	84.56	187.0	1532.0	15.38	0	8.30
STID	0.393	0.759	19.67	16.68	25.22	66.17	20.54	34.13	32.86	4.42	19.48	57.04	188.9	1589.1	16.66	1	8.60
AGCRN	–	–	–	17.53	25.83	71.00	19.57	32.65	31.41	8.39	26.59	202.07	211.5	1847.0	16.95	0	11.68
MTGNN	0.394	0.757	19.96	16.08	24.17	64.32	19.62	32.58	30.70	4.74	20.31	57.01	201.3	1628.1	17.26	0	9.99
GTS	0.405	0.783	20.54	17.66	26.77	63.81	19.82	34.04	32.49	4.41	23.19	**31.37**	241.1	1933.6	19.59	1	11.94

H EFFICIENCY

We measure training throughput and maximum trainable batch size on three benchmarks spanning a range of network sizes, namely SD (716 sensors), UrbanEV (1,682 sensors), and GBA (2,352 sensors), against the three strong representative baselines: GWNet, PatchSTG, and MAGE. For each model we search batch size by doubling from 1 up to a cap of 64, taking the largest value that trains without an out-of-memory error on a single NVIDIA RTX A6000. Every configuration we tested fits comfortably within a single GPU's budget at these batch sizes. Training itself used the fixed BS $= 16$ protocol of Appendix F; the batch sizes reported here reflect only this memory-capacity search. MAE is reproduced from Table 1 for reference alongside the efficiency numbers.

Table 6: **Training efficiency on SD, GBA, and UrbanEV.** Max BS is capped at 64; Sec/epoch and Samp/s (throughput) are both measured at that batch size, mean over 3 epochs. **Best** and second best per column among the models compared here.

Model	SD				GBA				UrbanEV			
	MAE	BS	Sec/ep	Samp/s	MAE	BS	Sec/ep	Samp/s	MAE	BS	Sec/ep	Samp/s
GraFT (PatchTST)	16.64	64	325.47	64	**19.09**	32	1046.22	20	**3.81**	64	318.48	98
GraFT (MLP)	16.72	64	103.23	203	19.51	64	326.14	64	3.97	64	115.76	270
GraFT (TSMixer)	17.02	64	131.36	160	20.35	64	412.07	51	3.94	64	150.20	208
MAGE	**16.35**	64	**61.88**	**339**	19.84	64	**321.69**	**65**	3.88	64	**35.32**	**884**
PatchSTG	16.98	64	172.29	122	19.67	32	998.82	21	4.74	64	132.74	235
GWNet	18.07	64	191.90	109	20.83	64	990.56	21	4.50	64	244.98	128

Two patterns hold across all three benchmarks. First, on the largest graph tested (GBA, 2,352 sensors), GraFT (MLP) and GraFT (TSMixer) both sustain the full batch-size cap of 64, while PatchSTG and GraFT (PatchTST), whose attention-based components are memory-heavy, are limited to 32. The graph operator itself does not cause this: it never materializes the $N \times N$ adjacency (Proposition 5), so its memory cost grows with Nd_G, not N^2. Second, GraFT (MLP) trains faster than both GWNet and PatchSTG on every benchmark tested, and GraFT (TSMixer) does so on two of three (trailing PatchSTG on UrbanEV); MAGE remains the fastest model overall, which is expected rather than a weakness of GraFT's design, since MAGE's single shared-expert transformer backbone has no separate per-sensor temporal predictor to run before its own graph mechanism. Excluding MAGE, GraFT (MLP) is consistently the fastest.

The PatchTST backbone is the exception: it is GraFT's most accurate configuration (Table 1) but its slowest, since PatchTST's attention-based predictor is more expensive per step than the MLP or TSMixer alternatives. Because the refinement stack itself is unchanged across all three backbones, this is a choice available to a practitioner, not a limitation: MLP or TSMixer for an efficiency-constrained deployment, PatchTST when accuracy is the priority.

I DISCUSSIONS AND LIMITATIONS

Gated-substrate comparison. Our real-data ablations (Table 2) isolate the refinement stack (*no graph*), its gate (*no gating*), and the learned graph (*identity graph*), but do not include a substrate architecture gated by the same zero-initialized α_ℓ mechanism. Constructing such a comparison fairly across all three backbones is nontrivial: unlike STGCN- or GraphWaveNet-style architectures, PatchTST and TSMixer have no established point at which to interleave a graph operator mid-network, so a gated-substrate variant of either would require designing and independently tuning a new architecture instead of reusing a published baseline, introducing a confound (our own untested design choices) rather than isolating placement from gating.

Variance across seeds. All results are reported as means over 5 seeds (Appendix F); we do not additionally report standard deviations. A few benchmark margins in Table 1 are close (for example, PEMS08 and GLA), and we leave a fuller significance analysis to future work.

Small-gate regime at deployment. Corollary (b) of Proposition 4 (Section 3.3) is stated for gates satisfying $|\alpha_\ell| \ll 1/M_j$; the trained gate magnitudes we observe (Section 5.3, 0.28–1.18) are larger than this regime is meant to cover, so the simplified additive form is best read as a near-initialization intuition rather than a description of the deployed model. The bound we rely on empirically is the

unconditional one: exactly zero contamination when gates are fully closed (corollary (a)). It holds at any gate magnitude and is what our synthetic and real-world results test directly.

Foundation time-series backbones. We instantiate ϕ_θ with an MLP, PatchTST, and TSMixer, each trained from scratch on every dataset. Since the principle is backbone-agnostic (Section 3.4), a time-series foundation model, or an in-context Transformer predictor (Lu et al., 2025), could in principle serve as ϕ_θ beneath the same gated refinement stack; we leave this to future work.