LIGHTWEIGHT AND INTERPRETABLE TRANSFORMER VIA MIXED GRAPH ALGORITHM UNROLLING FOR TRAFFIC FORECAST

Anonymous authors
Paper under double-blind review

ABSTRACT

Unlike conventional "black-box" transformers with classical self-attention mechanism, we build a lightweight and interpretable transformer-like neural net by unrolling a mixed-graph-based optimization algorithm to forecast traffic with spatial and temporal dimensions. We construct two graphs: an undirected graph $\mathcal{G}^u$ capturing spatial correlations across geography, and a directed graph $\mathcal{G}^d$ capturing sequential relationships over time. We predict future samples of signal $\mathbf{x}$, assuming it is "smooth" with respect to both $\mathcal{G}^u$ and $\mathcal{G}^d$, where we design new ℓ_2 and ℓ_1-norm variational terms to quantify and promote signal smoothness (low-frequency reconstruction) on a directed graph. We design an iterative algorithm based on alternating direction method of multipliers (ADMM), and unroll it into a feed-forward network for data-driven parameter learning. We insert graph learning modules for $\mathcal{G}^u$ and $\mathcal{G}^d$ that play the role of self-attention. Experiments show that our unrolled networks achieve competitive traffic forecast performance as state-of-the-art prediction schemes, while reducing parameter counts drastically.

1 INTRODUCTION

Transformer, based on the classical self-attention mechanism (Vaswani et al., 2017), is now a dominant deep learning (DL) architecture that has achieved state-of-the-art (SOTA) performance across multiple fields (Woźniak et al., 2020; Zamir et al., 2022). However, like other "off-the-shelf" DL models, transformer has a massive number of parameters, and its internal structure is not easily interpretable. Parameter reduction is a major industrial concern for practical computation and/or memory-constrained environments[1]. *Algorithm unrolling* (Monga et al., 2021) offers an alternative hybrid model-based / data-driven paradigm; by "unrolling" iterations of a model-based algorithm minimizing a well-defined optimization objective into neural layers stacked together to form a feed-forward network for data-driven parameter learning, the resulting network can be *both* mathematically interpretable[2] *and* competitive in performance. Notably, Yu et al. (2023) designs an algorithm minimizing a sparse rate reduction (SRR) objective that unrolls into a transformer-like neural net for image classification. Thuc et al. (2024) designs graph-based algorithms minimizing graph smoothness priors that unroll into transformer-like neural nets for image interpolation while reducing parameters. However, only a positive *undirected* graph model was used in Thuc et al. (2024) to capture simple pairwise correlations between neighboring pixels in a static image.

Spatial-temporal data (*e.g.*, traffic and weather) contains more complex node-to-node relationships: geographically near stations exhibit *spatial correlations*, while past observations influences future data, resulting in *sequential relationships*. In this paper, we study the design of transformers via graph algorithm unrolling for spatial-temporal data, using traffic forecast as a concrete example application.

We roughly categorize traffic forecast methods into model-based and DL-based methods. Among model-based methods, one approach to traffic forecast is to filter the signal along the spatial and temporal dimensions independently (Ramakrishna et al., 2020)—*e.g.*, employ graph spectral filters in the *graph signal processing* (GSP) literature (Ortega et al., 2018; Cheung et al., 2018) along the

[1] See a brief review of lightweight learning models for different applications in Appendix A.1.

[2] Common in algorithm unrolling (Monga et al., 2021), "interpretability" here means that each neural layer corresponds to an iteration of an optimization algorithm minimizing a mathematically-defined objective.

irregular spatial dimension, and traditional Fourier filters along the regular time dimension. Another approach is to model the spatial correlations across time as a sequence of slowly time-varying graphs, and then filter each spatial signal at time t separately, where the changes in consecutive graph adjacency matrices in time are constrained by the Frobenius norm (Kalofolias et al., 2017), ℓ_1-norm (Yamada et al., 2019), or low-rankness (Bagheri et al., 2024). However, neither approach exploits spatial and temporal relationships simultaneously for optimal performance.

Recent DL-based efforts address traffic forecast by building elaborate transformer architectures to capture spatial and temporal relations (Feng & Tassiulas, 2022; Gao et al., 2022; Jin et al., 2024)[3]. Like transformers used in other fields (Zamir et al., 2022), these are also black boxes with large parameter counts. *Graph attention networks* (GAT) (Velickovic et al., 2018) and *graph transformers* (Dwivedi & Bresson, 2021) are adaptations of the self-attention mechanism in transformers for graph-structured data (*e.g.*, computing output embeddings using only input embeddings in local graph neighborhoods). However, they are still uninterpretable and maintain large parameter sizes.

In contrast, we extend Thuc et al. (2024) to build lightweight "white-box" transformers that are fully interpretable via *mixed*-graph algorithm unrolling for traffic forecast. Specifically, we learn *two* graphs from data: i) *undirected* graph $\mathcal{G}^u$ to capture spatial correlations across geography, and ii) a *directed* graph $\mathcal{G}^d$ to capture sequential relationships over time. We show that our graph learning modules play the role of self-attention (Bahdanau et al., 2014), and thus our unrolled graph-based neural nets are transformers. Given the two learned graphs, we define a prediction objective for future samples in signal $\mathbf{x}$, assuming $\mathbf{x}$ is "smooth" with respect to (w.r.t.) both $\mathcal{G}^u$ and $\mathcal{G}^d$ in variational terms: *graph Laplacian regularizer* (GLR) (Pang & Cheung, 2017) for undirected $\mathcal{G}^u$, and newly designed *directed graph Laplacian regularizer* (DGLR) and *directed graph total variation* (DGTV) for directed $\mathcal{G}^d$. We design a corresponding linear-time optimization algorithm based on *alternating direction method of multipliers* (ADMM) (Boyd et al., 2011) that unrolls into neural layers for parameter learning. **For the first time, our DGLR and DGTV variational terms quantify and promote signal smoothness on a directed graph, with intuitive low-pass filter interpretations**[4]. Experiments show that our networks achieve competitive traffic forecast performance as SOTA prediction schemes, while employing drastically fewer parameters (**7.2% of transformer-based PDFormer** (Jiang et al., 2023)).

2 PRELIMINARIES

Undirected Graph Definitions: Denote by $\mathcal{G}^u(\mathcal{V}, \mathcal{E}^u, \mathbf{W}^u)$ an *undirected* graph with node set $\mathcal{V} = \{1, \ldots, N\}$ and undirected edge set $\mathcal{E}^u$, where $(i,j) \in \mathcal{E}^u$ implies that an undirected edge exists connecting nodes i and j with weight $w^u_{i,j} = W^u_{i,j}$, and $(i,j) \notin \mathcal{E}$ implies $W^u_{i,j} = 0$. $\mathbf{W}^u \in \mathbb{R}^{N \times N}$ is the symmetric *adjacency matrix* for $\mathcal{G}^u$. Denote by $\mathbf{D}^u \in \mathbb{R}^{N \times N}$ a diagonal *degree matrix*, where $D^u_{i,i} = \sum_j W^u_{j,i}$. The symmetric *graph Laplacian matrix* is $\mathbf{L}^u \triangleq \mathbf{D}^u - \mathbf{W}^u$ (Ortega et al., 2018). $\mathbf{L}^u$ is *positive semi-definite* (PSD) if all edge weights are non-negative $w^u_{i,j} \geq 0, \forall i,j$ (Cheung et al., 2018). The symmetric *normalized graph Laplacian* matrix is $\mathbf{L}^u_n \triangleq \mathbf{I} - (\mathbf{D}^u)^{-1/2}\mathbf{W}^u(\mathbf{D}^u)^{-1/2}$, while the asymmetric *random-walk graph Laplacian* matrix is $\mathbf{L}^u_r \triangleq \mathbf{I} - (\mathbf{D}^u)^{-1}\mathbf{W}^u$.

One can define a *graph spectrum* by eigen-decomposing the real and symmetric graph Laplacian $\mathbf{L}^u$ (or normalized graph Laplacian $\mathbf{L}^u_n$), where the k-th eigen-pair $(\lambda_k, \mathbf{v}_k)$ is interpreted as the k-th graph frequency and Fourier mode, respectively (Ortega et al., 2018). Given $\{\mathbf{v}_k\}$ are orthonormal vectors, one can decompose a signal $\mathbf{x}$ into its graph frequency components as $\boldsymbol{\alpha} = \mathbf{V}^\top \mathbf{x}$, where $\mathbf{L}^u = \mathbf{V}\text{diag}(\{\lambda_k\})\mathbf{V}^\top$, and $\mathbf{V}^\top$ is the *graph Fourier transform* (GFT).

The ℓ_2-norm *graph Laplacian regularizer* (GLR) (Pang & Cheung, 2017), $\mathbf{x}^\top \mathbf{L}^u \mathbf{x} = \sum_{(i,j) \in \mathcal{E}^u} w^u_{i,j}(x_i - x_j)^2$, is used to regularize an ill-posed signal restoration problem, like denoising, to bias low-frequency signal reconstruction (*i.e.*, signals consistent with similarity graph $\mathcal{G}^u$)

[3] A detailed review of DL models on traffic forecast is included in Appendix A.2.

[4] See a detailed review of related works on directed graph Laplacians and discussion of our novelty in directed graph frequency analysis in Appendix A.3.

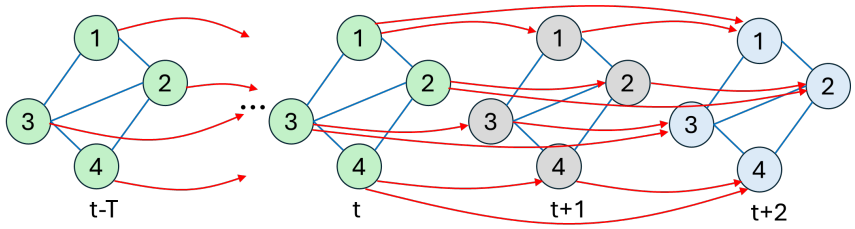


Figure 1: Example of a mixed graph with undirected edges (blue) connecting nodes of the same time instants, and directed edges (red) connecting nodes at t to nodes in window $\{t+1, t+2\}$.

given observation $\mathbf{y}$:

$$\begin{aligned}\mathbf{x}^* &= \arg\min_{\mathbf{x}} \|\mathbf{y}-\mathbf{x}\|_2^2 + \mu\,\mathbf{x}^\top \mathbf{L}^u \mathbf{x} \\ &= (\mathbf{I}+\mu\mathbf{L}^u)^{-1}\mathbf{y} = \mathbf{V}\mathrm{diag}\left(\frac{1}{1+\mu\lambda_1},\ldots,\frac{1}{1+\mu\lambda_N}\right)\mathbf{V}^\top\mathbf{y}. \end{aligned} \tag{1}$$

μ is a bias-variance tradeoff parameter, and the low-pass filter response is $f(\lambda) = (1+\mu\lambda)^{-1}$.

Directed Graph Definitions: We define analogous notations for a *directed* graph, denoted by $\mathcal{G}^d(\mathcal{V}, \mathcal{E}^d, \mathbf{W}^d)$. $[i,j] \in \mathcal{E}^d$ implies a directed edge exists from node i to j with weight $w^d_{j,i} = W^d_{j,i}$. $\mathbf{W}^d \in \mathbb{R}^{N\times N}$ is the asymmetric *adjacency matrix*. Denote by $\mathbf{D}^d \in \mathbb{R}^{N\times N}$ a diagonal *in-degree matrix*, where $D^d_{i,i} = \sum_j W^d_{i,j}$. The *directed graph Laplacian* matrix is defined as $\mathbf{L}^d \triangleq \mathbf{D}^d - \mathbf{W}^d$. The *directed random-walk graph Laplacian* matrix is $\mathbf{L}^d_r \triangleq \mathbf{I} - (\mathbf{D}^d)^{-1}\mathbf{W}^d$.

Similar frequency notion cannot be simply defined for directed graphs via eigen-decomposition of directed graph Laplacian $\mathbf{L}^d$, because $\mathbf{L}^d$ is asymmetric, and thus the Spectral Theorem (Hawkins, 1975) does not apply. We circumvent this problem by designing new variational terms in Section 3.2.

3 Optimization Formulation & Algorithm

3.1 Mixed Graph for Spatial/Temporal Data

We describe the construction of a mixed graph for spatial/temporal data. A measurement station i observes samples $x_i^{t-T}, \ldots, x_i^t$ at past and present instants $t-T, \ldots, t$ and samples $x_i^{t+1}, \ldots, x_i^{t+S}$ at future instants $t+1, \ldots, t+S$. Denote by $\mathbf{x} = [\mathbf{x}^{t-T}; \ldots; \mathbf{x}^{t+S}] \in \mathbb{R}^{N(T+S+1)}$ the target graph signal—a concatenation of samples from all N nodes across all $T+S+1$ instants. A *product graph* $\mathcal{G}$ of $N \times (T+S+1)$ nodes represents samples at $T+S+1$ instants. $\mathcal{G}$ is a mixture of: i) undirected graph $\mathcal{G}^u$ connecting spatially node pairs of the same instants, and ii) directed graph $\mathcal{G}^d$ connecting temporally each node i at instant τ to the same node at instants $\tau+1, \ldots, \tau+W$, where W denotes the pre-defined *time window*. See Fig. 1 for an illustration of a mixed graph with undirected spatial edges (blue) and directed temporal edges (red) for $W = 2$.

3.2 Variational Terms for Directed Graphs

Our constructed $\mathcal{G}^d$ is a *directed acyclic graph* (DAG); a directed edge always stems from a node at instant τ to a node at a future instant $\tau+s$, and thus no cycles exist. We first define a symmetric variational term for $\mathcal{G}^d$ called *directed graph Laplacian regularizer* (DGLR) to quantify variation of a signal $\mathbf{x}$ on $\mathcal{G}^d$. Denote by $\mathcal{S} \subset \mathcal{N}$ the set of *source nodes* with zero in-degrees, and $\bar{\mathcal{S}} \triangleq \mathcal{N} \setminus \mathcal{S}$ its complement. First, we add a self-loop of weight 1 to each node in $\mathcal{S}$; this ensures that $(\mathbf{D}^d)^{-1}$ is well-defined. Next, we define the row-stochastic *random-walk adjacency* matrix $\mathbf{W}^d_r \triangleq (\mathbf{D}^d)^{-1}\mathbf{W}^d$.

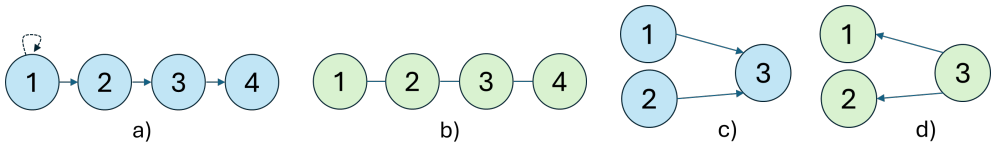


Figure 2: Example of a 4-node DAG $\mathcal{G}^d$, with an added self-loop at node 1, specified by $\mathbf{L}_r^d$ (a), and corresponding undirected graph $\mathcal{G}^u$, specified by $\mathcal{L}_r^d = (\mathbf{L}_r^d)^\top \mathbf{L}_r^d = \mathbf{L}^u$ (b). Example of a 3-node DAG (c), and a 3-node DAG with opposite directional edges (d).

3.2.1 ℓ_2-NORM VARIATIONAL TERM

Interpreting $\mathbf{W}_r^d$ as a *graph shift operator* (GSO) (Chen et al., 2015) on $\mathcal{G}^d$, we first define an ℓ_2-norm variational term[5] for $\mathcal{G}^d$ as the squared difference between $\mathbf{x}$ and its graph-shifted version $\mathbf{W}_r^d\mathbf{x}$:

$$\|\mathbf{x} - \mathbf{W}_r^d\mathbf{x}\|_2^2 = \|(\mathbf{I} - \mathbf{W}_r^d)\mathbf{x}\|_2^2 = \mathbf{x}^\top \underbrace{(\mathbf{L}_r^d)^\top \mathbf{L}_r^d}_{\mathcal{L}_r^d} \mathbf{x} \tag{2}$$

where *symmetrized directed graph Laplacian* matrix $\mathcal{L}_r^d \triangleq (\mathbf{L}_r^d)^\top \mathbf{L}_r^d$ is symmetric and PSD. We call $\mathbf{x}^\top \mathcal{L}_r^d \mathbf{x}$ the DGLR, which computes the sum of squared differences between each child $j \in \bar{\mathcal{S}}$ and its parents i for $[i, j] \in \mathcal{E}^d$. Note that DGLR computes to zero for the constant vector $\mathbf{1}$:

$$\mathbf{1}^\top \mathcal{L}_r^d \mathbf{1} = \mathbf{1}^\top (\mathbf{L}_r^d)^\top \mathbf{L}_r^d \mathbf{1} = \mathbf{1}^\top (\mathbf{L}_r^d)^\top (\mathbf{I} - \mathbf{W}_r^d)\mathbf{1} \overset{(a)}{=} \mathbf{0}$$

where (a) follows from $\mathbf{W}_r^d$ being row-stochastic. This makes sense, as constant $\mathbf{1}$ is the smoothest signal and has no variation across any graph kernels.

We interpret eigen-pairs $\{(\xi_k, \mathbf{u}_k)\}$ of $\mathcal{L}_r^d$ as graph frequencies and graph Fourier modes for directed graph $\mathcal{G}^d$, respectively, similar to eigen-pairs $\{(\lambda_k, \mathbf{v}_k)\}$ of graph Laplacian $\mathbf{L}^u$ for an undirected graph. Importantly, we prove that our frequency definition using DGLR defaults to pure sinusoids as frequencies in the unweighted directed line graph case. Thus, our algorithm minimizing DGLR includes GSP schemes like Ramakrishna et al. (2020) that analyze signals along the time dimension using classical Fourier filters as special cases.

Theorem 3.1. *Consider a directed line graph $\mathcal{G}^d$ of N nodes with directed edge weights equal to 1, where the first (source) node is augmented with a self-loop of weight 1. The symmetrized directed graph Laplacian $\mathcal{L}_r^d = (\mathbf{L}_r^d)^\top \mathbf{L}_r^d$, where $\mathbf{L}_r^d$ is the random-walk graph Laplacian for $\mathcal{G}^d$, is the same as graph Laplacian $\mathbf{L}^u$ for an undirected line graph $\mathcal{G}^u$ of N nodes with edge weights equal to 1.*

See Appendix B for a formal proof.

Remark: Theorem 3.1 only gives a special case when the DGLR of a graph is equal to the GLR of its undirected base graph. In fact, **using our symmetrized $\mathcal{L}_r^d$ does not mean directionality has been lost.** As an example, for the 3-node DAG in Fig. 2(c), the variational term is $\|\mathbf{L}_r^d\mathbf{x}\|_2^2 = (x_3 - \frac{1}{2}(x_1 + x_2))^2$ and computes to 0 for $\mathbf{x} = [2\ \ 0\ \ 1]^\top$. On the other hand, for the DAG in Fig. 2(d) with opposite directions, the variational term is $\|\mathbf{L}_r^d\mathbf{x}\|_2^2 = (x_1 - x_3)^2 + (x_2 - x_3)^2$ and computes to 0 only if $\mathbf{x}$ is a constant vector. With our directed graph contruction with *time windows* as described in Section 3.1, since all nodes after the second time step has more than one predecessors, the DGLR of the undirected graph does not equal the GLR of its base graph.

3.2.2 ℓ_1-NORM VARIATIONAL TERM

Using $\mathbf{W}_r^d$ again as a GSO, we define an ℓ_1-norm variational term called *directed graph total variation* (DGTV) as

$$\|\mathbf{x} - \mathbf{W}_r^d\mathbf{x}\|_1 = \|\mathbf{L}_r^d\mathbf{x}\|_1 = \sum_{j \in \bar{\mathcal{S}}} |x_j - \sum_i w_{j,i} x_i|. \tag{3}$$

Unlike DGLR in eq. (2), DGTV is not symmetric and has no obvious frequency interpretation. Nonetheless, we provide a *two-channel filterbank* interpretation in the sequel.

[5] A similar directed graph variational term was defined in Li et al. (2023) towards an objective for directed graph sampling. We focus instead on directed graph signal restoration.

3.3 Optimization Formulation

Denote by $\mathbf{y} \in \mathbb{R}^M$ the observation, $\mathbf{x} \in \mathbb{R}^{N(T+S+1)}$ the target signal, and $\mathbf{H} \in \{0,1\}^{M \times N(T+S+1)}$ the sampling matrix that selects M observed samples from $N(T+S+1)$ entries in $\mathbf{x}$. Given defined undirected and directed edges, we write an objective for $\mathbf{x}$ containing a squared-error fidelity term and graph smoothness terms for the two graphs $\mathcal{G}^u$ and $\mathcal{G}^d$—GLR, DGLR, and DGTV:

$$\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \mu_u \mathbf{x}^\top \mathbf{L}^u \mathbf{x} + \mu_{d,2} \mathbf{x}^\top \mathcal{L}_r^d \mathbf{x} + \mu_{d,1} \|\mathbf{L}_r^d \mathbf{x}\|_1 \tag{4}$$

where $\mu_u, \mu_{d,2}, \mu_{d,1} \in \mathbb{R}_+$ are weight parameters for the three regularization terms. Combination of ℓ_2- and ℓ_1-norm penalties—DGLR and DGTV for directed graph $\mathcal{G}^d$ in our case[6]—is called *elastic net regularization* in statistics (Zou & Hastie, 2005) with demonstrable improved robustness. Further, employing both regularization terms means that, after algorithm unrolling, we can adapt an appropriate mixture by learning weights $\mu_{d,2}$ and $\mu_{d,1}$ per neural layer.

Convex minimization in eq. (4) is composed of smooth ℓ_2-norm terms and one non-smooth ℓ_1-norm term. We pursue a divide-and-conquer approach and solve eq. (4) via an ADMM framework (Boyd et al., 2011) by minimizing ℓ_2- and ℓ_1-norm terms in eq. (4) alternately.

3.4 ADMM Optimization Algorithm

We first introduce *auxiliary variable* $\boldsymbol{\phi}$ and rewrite the optimization in eq. (4) as

$$\min_{\mathbf{x}, \boldsymbol{\phi}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \mu_u \mathbf{x}^\top \mathbf{L}^u \mathbf{x} + \mu_{d,2} \mathbf{x}^\top \mathcal{L}_r^d \mathbf{x} + \mu_{d,1} \|\boldsymbol{\phi}\|_1, \qquad \text{s.t.} \quad \boldsymbol{\phi} = \mathbf{L}_r^d \mathbf{x}. \tag{5}$$

Using the augmented Lagrangian method (Boyd & Vandenberghe, 2004), we rewrite eq. (5) in an unconstrained form:

$$\min_{\mathbf{x}, \boldsymbol{\phi}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \mu_u \mathbf{x}^\top \mathbf{L}^u \mathbf{x} + \mu_{d,2} \mathbf{x}^\top \mathcal{L}_r^d \mathbf{x} + \mu_{d,1} \|\boldsymbol{\phi}\|_1 + \boldsymbol{\gamma}^\top (\boldsymbol{\phi} - \mathbf{L}_r^d \mathbf{x}) + \frac{\rho}{2} \|\boldsymbol{\phi} - \mathbf{L}_r^d \mathbf{x}\|_2^2 \tag{6}$$

where $\boldsymbol{\gamma} \in \mathbb{R}^{N(T+S+1)}$ is a Lagrange multiplier vector, and $\rho \in \mathbb{R}_+$ is a non-negative ADMM parameter. We solve eq. (6) iteratively by minimizing $\mathbf{x}$ and $\boldsymbol{\phi}$ in alternating steps till convergence.

3.4.1 Minimizing $\mathbf{x}^{\tau+1}$

Fixing $\boldsymbol{\phi}^\tau$ and $\boldsymbol{\gamma}^\tau$ at iteration τ, optimization in eq. (6) for $\mathbf{x}^{\tau+1}$ becomes an unconstrained convex quadratic objective. The solution is a system of linear equations:

$$\left(\mathbf{H}^\top \mathbf{H} + \mu_u \mathbf{L}^u + (\mu_{d,2} + \frac{\rho}{2}) \mathcal{L}_r^d\right) \mathbf{x}^{\tau+1} = (\mathbf{L}_r^d)^\top (\frac{\rho}{2} \boldsymbol{\phi}^\tau + \frac{\boldsymbol{\gamma}^\tau}{2}) + \mathbf{H}^\top \mathbf{y}. \tag{7}$$

Because coefficient matrix $\mathbf{H}^\top \mathbf{H} + \mu_u \mathbf{L}^u + (\mu_{d,2} + \frac{\rho}{2}) \mathcal{L}_r^d$ is sparse, symmetric and PD, eq. (7) can be solved in linear time via *conjugate gradient* (CG) (Shewchuk, 1994) without matrix inversion.

Splitting ℓ_2-norm Terms: Instead of solving eq. (7) directly for $\mathbf{x}^{\tau+1}$ at iteration τ, we introduce again auxiliary variables $\mathbf{z}_u$ and $\mathbf{z}_d$ in eq. (5) for the ℓ_2-norm GLR and DGLR terms respectively, and rewrite the optimization as

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{z}_u, \mathbf{z}_d} \quad & \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \mu_u \mathbf{z}_u^\top \mathbf{L}^u \mathbf{z}_u + \mu_{d,2} \mathbf{z}_d^\top \mathcal{L}_r^d \mathbf{z}_d + (\boldsymbol{\gamma}^\tau)^\top (\boldsymbol{\phi}^\tau - \mathbf{L}_r^d \mathbf{x}) + \frac{\rho}{2} \|\boldsymbol{\phi}^\tau - \mathbf{L}_r^d \mathbf{x}\|_2^2 \\ \text{s.t.} \quad & \mathbf{x} = \mathbf{z}_u = \mathbf{z}_d. \end{aligned} \tag{8}$$

Using again the augmented Lagrangian method, we rewrite the unconstrained version as

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{z}_u, \mathbf{z}_d} \; & \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \mu_u \mathbf{z}_u^\top \mathbf{L}^u \mathbf{z}_u + \mu_{d,2} \mathbf{z}_d^\top \mathcal{L}_r^d \mathbf{z}_d + (\boldsymbol{\gamma}^\tau)^\top (\boldsymbol{\phi}^\tau - \mathbf{L}_r^d \mathbf{x}) + \frac{\rho}{2} \|\boldsymbol{\phi}^\tau - \mathbf{L}_r^d \mathbf{x}\|_2^2 \\ & + (\boldsymbol{\gamma}_u^\tau)^\top (\mathbf{x} - \mathbf{z}_u) + \frac{\rho_u}{2} \|\mathbf{x} - \mathbf{z}_u\|_2^2 + (\boldsymbol{\gamma}_d^\tau)^\top (\mathbf{x} - \mathbf{z}_d) + \frac{\rho_d}{2} \|\mathbf{x} - \mathbf{z}_d\|_2^2 \end{aligned} \tag{9}$$

[6]We can employ both ℓ_2- and ℓ_1-norm regularization terms for the undirected graph $\mathcal{G}^u$ as well. For simplicity, we focus on designing new regularization terms for directed graphs in this paper.

where $\boldsymbol{\gamma}_u, \boldsymbol{\gamma}_d \in \mathbb{R}^{N(T+S+1)}$ are multipliers, and $\rho_u, \rho_d \in \mathbb{R}_+$ are ADMM parameters. Optimizing $\mathbf{x}$, $\mathbf{z}_u$ and $\mathbf{z}_d$ in eq. (9) in turn at iteration τ lead to three linear systems:

$$\left(\mathbf{H}^\top\mathbf{H} + \frac{\rho}{2}\mathcal{L}_r^d + \frac{\rho_u+\rho_d}{2}\mathbf{I}\right)\mathbf{x}^{\tau+1} = (\mathbf{L}_r^d)^\top\left(\frac{\boldsymbol{\gamma}^\tau}{2} + \frac{\rho}{2}\boldsymbol{\phi}^\tau\right) - \frac{\boldsymbol{\gamma}_u^\tau}{2} + \frac{\rho_u}{2}\mathbf{z}_u^\tau - \frac{\boldsymbol{\gamma}_d^\tau}{2} + \frac{\rho_d}{2}\mathbf{z}_d^\tau + \mathbf{H}^\top\mathbf{y} \tag{10}$$

$$\left(\mu_u\mathbf{L}^u + \frac{\rho_u}{2}\mathbf{I}\right)\mathbf{z}_u^{\tau+1} = \frac{\boldsymbol{\gamma}_u^\tau}{2} + \frac{\rho_u}{2}\mathbf{x}^{\tau+1} \tag{11}$$

$$\left(\mu_{d,2}\mathcal{L}_r^d + \frac{\rho_d}{2}\mathbf{I}\right)\mathbf{z}_d^{\tau+1} = \frac{\boldsymbol{\gamma}_d^\tau}{2} + \frac{\rho_d}{2}\mathbf{x}^{\tau+1}. \tag{12}$$

Thus, instead of solving eq. (7) for $\mathbf{x}^{\tau+1}$ directly at iteration τ, $\mathbf{x}^{\tau+1}$, $\mathbf{z}_u^{\tau+1}$ and $\mathbf{z}_d^{\tau+1}$ are optimized in turn via eq. (10) through eq. (12) using CG. This splitting induces more network parameters for data-driven learning after algorithm unrolling (see Section 4.1) towards better performance, and affords us spectral filter interpretations of the derived linear systems.

Interpretation: To interpret solution $\mathbf{z}_u^{\tau+1}$ spectrally, we rewrite eq. (11) as

$$\mathbf{z}_u^{\tau+1} = \left(\frac{2\mu_u}{\rho_u}\mathbf{L}^u + \mathbf{I}\right)^{-1}\left(\frac{\boldsymbol{\gamma}_u}{\rho_u} + \mathbf{x}^{\tau+1}\right) = \mathbf{V}\mathrm{diag}\left(\left\{\frac{1}{1+\frac{2\mu_u}{\rho_u}\lambda_k}\right\}\right)\mathbf{V}^\top\left(\frac{\boldsymbol{\gamma}_u}{\rho_u} + \mathbf{x}^{\tau+1}\right) \tag{13}$$

where $\mathbf{L}^u = \mathbf{V}\mathrm{diag}(\{\lambda_k\})\mathbf{V}^\top$ is the eigen-decomposition. Thus, using frequencies defined by eigen-pairs $\{(\lambda_k, \mathbf{v}_k)\}$ of $\mathbf{L}^u$, $\mathbf{z}_u^{\tau+1}$ is the *low-pass* filter output of $\mathbf{x}^{\tau+1}$ offset by $\frac{\boldsymbol{\gamma}_u}{\rho_u}$, with frequency response $f(\lambda) = (1+\frac{2\mu_u}{\rho_u}\lambda)^{-1}$.

As done for $\mathbf{z}_u^{\tau+1}$, we can rewrite $\mathbf{z}_d^{\tau+1}$ in eq. (12) as

$$\mathbf{z}_d^{\tau+1} = \mathbf{U}\mathrm{diag}\left(\left\{\frac{1}{1+\frac{2\mu_{d,2}}{\rho_d}\xi_k}\right\}\right)\mathbf{U}^\top\left(\frac{\boldsymbol{\gamma}_d}{\rho_d} + \mathbf{x}^{\tau+1}\right) \tag{14}$$

where $\mathcal{L}_r^d = \mathbf{U}\mathrm{diag}(\{\xi_k\})\mathbf{U}^\top$ is the eigen-decomposition. Analogously, $\mathbf{z}_d^{\tau+1}$ is the *low-pass* filter output of $\mathbf{x}^{\tau+1}$ offset by $\frac{\boldsymbol{\gamma}_d}{\rho_d}$, with frequency response $f(\xi) = (1+\frac{2\mu_{d,2}}{\rho_d}\xi)^{-1}$.

To interpret solution $\mathbf{x}^{\tau+1}$ in eq. (10) as a low-pass filter output, see Appendix C.

3.4.2 Minimizing $\boldsymbol{\phi}^{\tau+1}$

Fixing $\mathbf{x}^{\tau+1}$ at iteration τ, optimizing $\boldsymbol{\phi}^{\tau+1}$ in eq. (6) means

$$\boldsymbol{\phi}^{\tau+1} = \arg\min_{\boldsymbol{\phi}} g(\boldsymbol{\phi}) = \mu_{d,1}\|\boldsymbol{\phi}\|_1 + (\boldsymbol{\gamma}^\tau)^\top(\boldsymbol{\phi} - \mathbf{L}_r^d\mathbf{x}^{\tau+1}) + \frac{\rho}{2}\|\boldsymbol{\phi} - \mathbf{L}_r^d\mathbf{x}^{\tau+1}\|_2^2. \tag{15}$$

We can compute eq. (15) element-wise as follows (see Appendix D for a derivation):

$$\begin{aligned}\delta &= (\mathbf{L}_r^d)_i\mathbf{x}^{\tau+1} - \rho^{-1}\gamma_i^\tau \\ \phi_i^{\tau+1} &= \mathrm{sign}(\delta)\cdot\max(|\delta| - \rho^{-1}\mu_{d,1},\, 0).\end{aligned} \tag{16}$$

Interpretation: Soft-thresholding in eq. (16) to compute $\phi_i^{\tau+1}$ *attenuates* the i-th entry of $\mathbf{L}_r^d\mathbf{x}^{\tau+1}$. Given that graph Laplacians—second difference matrices on graphs—are high-pass filters (Ortega et al., 2018), we interpret $\mathbf{L}_r^d$ as the *high-pass channel* of a two-channel filterbank (Vetterli & Kovacevic, 2013), where the corresponding *low-pass channel* is $\mathbf{I} - \mathbf{L}_r^d = \mathbf{W}_r^d$. Using eq. (16) for processing means that the low-pass channel $\mathbf{W}_r^d$ is unused and thus preserved. For example, the constant (low-frequency) signal $\mathbf{1}$ passes through the low-pass channel undisturbed, *i.e.*, $\mathbf{1} = \mathbf{W}_r^d\mathbf{1}$, while it is filtered out completely by the high-pass channel, *i.e.*, $\mathbf{0} = \mathbf{L}_r^d\mathbf{1}$. Thus, attenuation of the high-pass channel means eq. (16) is a low-pass filter.

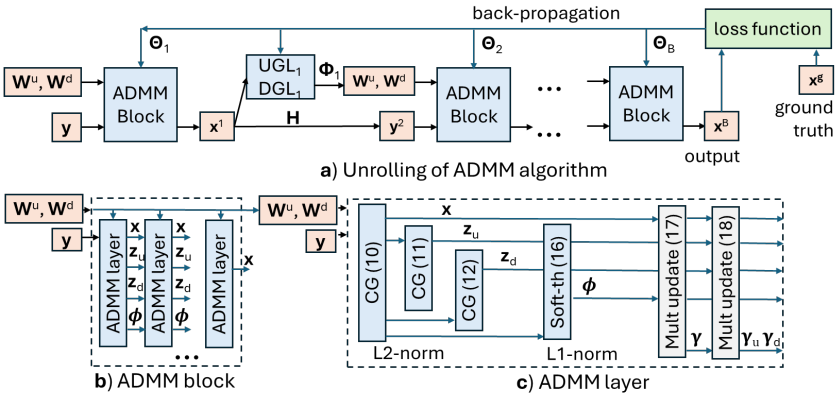


Figure 3: Unrolling of proposed iterative ADMM algorithm into blocks and neural layers.

3.4.3 Updating Lagrange Multipliers

For objective eq. (6), we follow standard ADMM procedure to update Lagrange multiplier $\boldsymbol{\gamma}^{\tau+1}$ after each computation of $\mathbf{x}^{\tau+1}$ and $\boldsymbol{\phi}^{\tau+1}$:

$$\boldsymbol{\gamma}^{\tau+1} = \boldsymbol{\gamma}^{\tau} + \rho(\boldsymbol{\phi}^{\tau+1} - \mathbf{L}_r^d \mathbf{x}^{\tau+1}). \tag{17}$$

Splitting the ℓ_2-norm terms means that additional multipliers $\boldsymbol{\gamma}_u^{\tau+1}$ and $\boldsymbol{\gamma}_d^{\tau+1}$ in objective eq. (9) must also be updated after each computation of $\mathbf{x}^{\tau+1}$, $\mathbf{z}_u^{\tau+1}$, $\mathbf{z}_d^{\tau+1}$, and $\boldsymbol{\phi}^{\tau+1}$:

$$\boldsymbol{\gamma}_u^{\tau+1} = \boldsymbol{\gamma}_u^{\tau} + \rho_u(\mathbf{x}^{\tau+1} - \mathbf{z}_u^{\tau+1}), \qquad \boldsymbol{\gamma}_d^{\tau+1} = \boldsymbol{\gamma}_d^{\tau} + \rho_d(\mathbf{x}^{\tau+1} - \mathbf{z}_d^{\tau+1}). \tag{18}$$

The ADMM loop is repeated till $\mathbf{x}^{\tau+1}$ and $\boldsymbol{\phi}^{\tau+1}$ converge.

4 Unrolled Neural Network

We unroll our iterative ADMM algorithm to an interpretable neural net. The crux to our network is two periodically inserted graph learning modules for undirected and directed edge weight computation that are akin to the classical self-attention mechanism in transformers (Bahdanau et al., 2014).

4.1 ADMM Algorithm Unrolling

We implement each iteration of our algorithm in Section 3.4 as a neural layer. Specifically, each ADMM iteration is implemented as four sub-layers in sequence: solving linear systems eq. (10) for $\mathbf{x}^{\tau+1}$, eq. (11) for $\mathbf{z}_u^{\tau+1}$, and eq. (12) for $\mathbf{z}_d^{\tau+1}$ via CG, and computing $\boldsymbol{\phi}^{\tau+1}$ via eq. (16). CG is itself an iterative descent algorithm, where parameters α and β corresponding to step size and momentum can be tuned end-to-end (see Appendix E.1 for detailed implementation). Weight parameters $\mu_u, \mu_{d,2}, \mu_{d,1}$ for prior terms and ADMM parameters ρ, ρ_u, ρ_d are also tuned per layer. Multipliers $\boldsymbol{\gamma}^{\tau+1}, \boldsymbol{\gamma}_u^{\tau+1}, \boldsymbol{\gamma}_d^{\tau+1}$ are then updated via eq. (17) and eq. (18) to complete one layer. Parameters learned during back-propagation for each ADMM block b are denoted by $\boldsymbol{\Theta}_b$. See Fig. 3 for an illustration, and Appendix E.2 for the detailed algorithms.

4.2 Self-Attention Operator in Transformer

We first review the basic self-attention mechanism: a scaled dot product following by a softmax operation (Bahdanau et al., 2014). Denote by $\mathbf{x}_i \in \mathbb{R}^E$ an *embedding* for token i. The *affinity* between tokens i and j, $e(i,j)$, is the dot product of linear-transformed $\mathbf{K}\mathbf{x}_i$ and $\mathbf{Q}\mathbf{x}_j$, where $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{E\times E}$ are the *query* and *key* matrices, respectively. Using softmax, non-negative *attention weight* $a_{i,j}$ is computed from $e(i,j)$'s as

$$a_{i,j} = \frac{\exp(e(i,j))}{\sum_{l=1}^{N} \exp(e(i,l))}, \qquad e(i,j) = (\mathbf{Q}\mathbf{x}_j)^{\top}(\mathbf{K}\mathbf{x}_i). \tag{19}$$

Given attention weights $a_{i,j}$, output embedding $\mathbf{y}_i$ for token i is computed as

$$\mathbf{y}_i = \sum_{l=1}^{N} a_{i,l} \mathbf{x}_l \mathbf{V} \tag{20}$$

where $\mathbf{V} \in \mathbb{R}^{E \times E}$ is a *value* matrix. By "self-attention", we mean that output embeddings are computed from weighted input embeddings. A transformer is thus a sequence of embedding-to-embedding mappings via self-attention operations defined by learned $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ matrices.

4.3 Graph Learning Modules

Undirected Graph Learning: We learn an undirected graph $\mathcal{G}^u$ in a module $\texttt{UGL}_b$ at block b. For each node i, we compute a low-dimensional feature vector $\mathbf{f}_i^u = F^u(\mathbf{e}_i) \in \mathbb{R}^K$, where $\mathbf{e}_i \in \mathbb{R}^E$ is a vector of signal values, position embeddings in time and Laplacian embeddings of the physical graph as done in Feng & Tassiulas (2022), and $F^u(\cdot)$ is a chosen learned nonlinear *feature function* $F^u : \mathbb{R}^E \mapsto \mathbb{R}^K$, *e.g.*, a shallow GNN (see Appendix F.3 for details). As done in Thuc et al. (2024), the feature distance between nodes i and j is computed as the *Mahalanobis distance* $\mathrm{d}^u(i,j)$:

$$\mathrm{d}^u(i,j) = (\mathbf{f}_i^u - \mathbf{f}_j^u)^\top \mathbf{M} (\mathbf{f}_i^u - \mathbf{f}_j^u) \tag{21}$$

where $\mathbf{M} \succeq 0$ is a symmetric PSD *metric matrix* of dimension $\mathbb{R}^{K \times K}$. Distance in eq. (21) is non-negative and *symmetric*, *i.e.*, $\mathrm{d}^u(i,j) = \mathrm{d}^u(j,i)$. Weight $w_{i,j}^u$ of undirected edge $(i,j) \in \mathcal{E}^u$ is computed as

$$w_{i,j}^u = \frac{\exp\left(-\mathrm{d}^u(i,j)\right)}{\sqrt{\sum_{l \in \mathcal{N}_i} \exp(-\mathrm{d}^u(i,l))} \sqrt{\sum_{k \in \mathcal{N}_j} \exp(-\mathrm{d}^u(k,j))}} \tag{22}$$

where the denominator summing over 1-hop neighborhood $\mathcal{N}_i$ is inserted for normalization.

Remark: Interpreting distance $\mathrm{d}^u(i,j)$ as $-e(i,j)$, edge weights $w_{i,j}^u$'s in eq. (22) are essentially attention weights $a_{i,j}$'s in eq. (19), and thus the undirected graph learning module constitutes a self-attention mechanism, albeit requiring fewer parameters, since parameters for function $F^u(\cdot)$ and metric matrix $\mathbf{M}$ can be much smaller than large and dense query and key matrices, $\mathbf{Q}$ and $\mathbf{K}$. Further, no value matrix $\mathbf{V}$ is required, since output signal $\mathbf{x}^{\tau+1}$ is computed through a set of low-pass filters—eq. (10), eq. (11), eq. (12) and eq. (16)—derived from our optimization of objective eq. (4).

Directed Graph Learning: Similarly, we learn a directed graph $\mathcal{G}^d$ in a module $\texttt{DGL}_b$ at block b. For each node i, we compute a feature vector $\mathbf{f}_i^d = F^d(\mathbf{e}_i) \in \mathbb{R}^K$ using feature function $F^d(\cdot)$. Like eq. (21), the Mahalanobis distance $\mathrm{d}^d(i,j)$ between nodes i and j is computed from $\mathbf{f}_i^d$ and $\mathbf{f}_j^d$ with a learned PSD metric $\mathbf{P}$. Weight $w_{j,i}^d$ of directed edge $[i,j] \in \mathcal{E}^d$ is then computed using exponentials and normalization as

$$w_{j,i}^d = \frac{\exp(-\mathrm{d}^d(j,i)}{\sum_{(i,k) \in \mathcal{E}^d} \exp(-\mathrm{d}^d(k,i))}. \tag{23}$$

Parameters learned via back-propagation by the two graph learning modules at block b—parameters of $F^u(\cdot)$, $F^d(\cdot)$, $\mathbf{M}$ and $\mathbf{P}$—are denoted by $\mathbf{\Phi}_b$.

5 Experiments

5.1 Experimental Setup

We evaluate our model's performance on commonly used traffic speed dataset METR-LA (Li et al., 2017) and traffic flow dataset PEMS03 (Guo et al., 2022), each of which contains city-size traffic data with a sampling interval of 5 minutes in a local region, together with the real connections and distances (or travel cost) between sensors. We reduce the size of the original dataset by sub-sampling 1/3 of all data uniformly [7], and split it into training, validation, and test sets by 6:2:2. We predict the

[7] With a drastically smaller parameter count, our model focuses on the efficiency in utilizing the observations and the effectiveness with small training sets. Uniform subsampling reduces the size of the training set while making full use of the entire observation. See a detailed analysis in Appendix G.4.

Table 1: Comparison of categorized baselines and model parameters for 60-minute forecasting on the PEMS03 dataset against ours.

Category	Selected Models	Params #
Model-based	VAR (Reinsel, 2003)	-
GNNs	STGCN†(Yu et al., 2018), STSGCN (Song et al., 2020)	321K, 3,496K
GATs	GMAN (Zheng et al., 2020), ST-Wave (Fang et al., 2023)	210K, 883K
Transformers	PDFormer (Jiang et al., 2023), STAEformer (Liu et al., 2023)	531K, 1,404K
Adaptive-graph models	Graph WaveNet (GWN) (Wu et al., 2019), AGCRN (Bai et al., 2020)	277K, 749K
MLP-based models	STID (Shao et al., 2022), SimpleTM (Chen et al., 2025)	123K, 540K
Mixed-graph unrolling	Ours	**38K**

† STGCN predicts only one time step at a time. We train the model to predict only the next step, and predict the entire sequence recursively.

Table 2: Comparison of RMSE / MAE / MAPE(%) metrics of our lightweight transformer to baseline models on 30 / 60-minute forecasting in PeMS03 and METR-LA datasets. We use **boldface** for the smallest error, color the 2nd and 3nd small error in **blue**, and underline the next 2 smallest errors.

Dataset & Horizon	PEMS03 (358 nodes, 547 edges)			METR-LA (207 nodes, 1,315 edges)		
	30 minutes	60 minutes	120 minutes	30 minutes	60 minutes	120 minutes
VAR	28.07 / 16.53 / 17.49	30.54 / 18.31 / 19.61	36.65 / 22.40 / 24.64	10.72 / 5.55 / 11.29	12.59 / 6.99 / 13.46	**14.83** / 8.90 / 16.54
STGCN	28.06 / 18.24 / 17.76	37.31 / 24.29 / 23.00	54.34 / 34.83 / 30.52	11.26 / 5.08 / 10.93	13.91 / 6.35 / 13.67	16.92 / 8.11 / 17.69
STSGCN	26.41 / 16.75 / 16.86	30.62 / 19.35 / 19.15	38.04 / 23.86 / 23.62	10.25 / 4.05 / 9.18	12.65 / 5.18 / 11.48	15.74 / **6.84** / 15.33
GMAN	25.79 / 16.23 / 20.04	27.57 / 17.48 / 24.33	**30.69** / 19.20 / 26.69	11.97 / 5.34 / 10.66	14.49 / 6.93 / 13.12	15.53 / 7.59 / 14.70
ST-Wave	25.57 / **15.11** / **15.04**	28.65 / 16.81 / 19.24	**29.88** / **17.11** / **17.71**	10.81 / 4.11 / 9.16	13.24 / 5.33 / 11.32	23.18 / 11.22 / **12.71**
PDFormer	**23.71** / **15.05** / 18.16	**27.16** / 17.26 / 21.21	35.77 / 22.25 / 25.01	10.21 / **3.89** / 8.50	**12.17** / **4.81** / **10.92**	17.27 / 9.55 / 19.10
STAEformer	30.22 / 18.85 / 26.62	38.36 / 23.68 / 29.21	48.72 / 31.96 / 44.71	10.16 / **3.73** / **8.25**	12.58 / **4.79** / **10.08**	**14.63** / **5.82** / **11.87**
GWN	25.76 / 15.22 / 16.83	28.15 / 18.11 / **17.56**	34.60 / 21.10 / 22.45	11.42 / 5.18 / **8.21**	12.46 / 5.17 / 11.97	23.13 / 11.32 / 13.53
AGCRN	27.40 / 15.19 / **14.35**	29.90 / **16.76** / **15.32**	32.79 / **18.72** / **17.03**	**10.11** / **3.82** / 8.61	12.56 / **5.00** / 11.25	**14.77** / **6.33** / 14.05
STID	26.50 / 17.27 / 18.59	31.88 / 20.82 / 24.89	42.98 / 28.03 / 42.41	10.21 / 4.05 / 9.47	12.61 / 5.21 / 12.22	15.48 / 6.98 / 16.78
SimpleTM	**23.75** / **15.13** / **15.59**	**25.56** / **15.97** / **15.49**	**30.58** / **18.73** / **18.18**	**8.76** / 4.12 / **7.63**	**11.96** / 5.49 / **9.65**	15.44 / 7.64 / **12.27**
Ours	**25.05** / 15.85 / 16.49	**26.96** / **16.58** / 18.07	34.06 / 20.23 / 23.86	**10.06** / 4.05 / 9.23	**12.17** / 5.27 / 11.78	15.19 / 7.14 / 16.44

traffic speed/flow in the following 30 / 60 /120 minutes (6 / 12 /24 steps) from the previously observed 60 minutes (12 steps) for all datasets. We restrict our model to learn a *sparse* graph based on the real road connections and a limited time window to model only the *local* spatial/temporal influence. For undirected graphs $\mathcal{G}^u$, we connect each node to its k nearest neighbors (NNs). We construct 5 ADMM blocks, each contains 25 ADMM layers, and insert a pair of graph learning modules before each ADMM block. We set feature dimensions $K = 6$, and stack vertically 4 graph learning modules to learn in parallel as an implementation of the *multi-head* attention mechanism. We minimize the Huber loss with $\delta = 1$ for the *entire* reconstructed sequence and the groundtruths. See Appendix F for more details.

5.2 Experimental Results

We train each run of our model on an entire NVIDIA GeForce RTX 3090. We evaluate our unrolling model against the baselines shown in Table 1. Each model is trained for 70 epochs on the reduced dataset, and the metrics are the Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) in the predicted 6 / 12 / 24 steps.

Table 2 shows the performance of our models and baselines in traffic forecasting, and the last column of Table 1 compares the number of parameters. We observe that our models achieve comparable performance to most baseline models, achieving top-3 performance in at least one metric for 30- and 60-minute forecasting, while employing drastically fewer parameters (**7.2% of transformer-based PDFormer**). Among all the models, ST-Wave and SimpleTM generally achieve the best performance. However, ST-Wave relies on an application-specific usage of wavelet transforms, and SimpleTM employs linear models with no inductive bias, which is parameter-heavy (Battaglia et al., 2018) and lacks interpretability; both models require even more parameters than PDFormer. The largest GCN model, STSGCN, fails to achieve a satisfying performance for PEMS03, where the real connections are highly sparse. STAEformer, the largest transformer, also overfits heavily in the reduced PEMS03 datasets. We also observe that our unrolled model

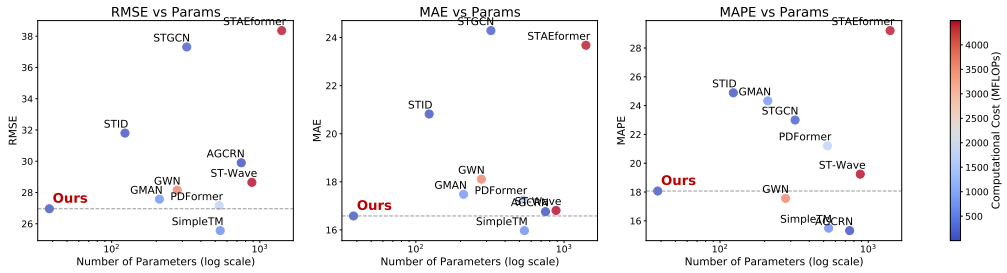


Figure 4: The trade-off between the number of parameters and performance for 60-minute forecast on PEMS03 dataset. The color of each scatter point represents the inference computational cost (in MFLOPs) in Table 6 in Appendix G.3.

also generally outperforms the multi-attention model GMAN. Though both our model and GMAN leverage local graph neighborhoods when computing attention, we dramatically reduce parameters by replacing classical self-attention mechanisms in transformers with unrolled ADMM iterations with no observable degradation in performance. *In summary, none of the DL-based baseline schemes unambiguously achieves the best performance across the two datasets, while all require significantly more parameters than our models. In contrast, our model achieves top-3 forecasting performance in two of three prediction windows for both datasets (top-5 in the remaining prediction window), in at least one metric.*

We plot the metrics with respect to the parameter size and computational cost for 60-minute forecast on the PEMS03 datasets in Figure 4. The conventional baseline models with larger parameter counts generally exhibit lower prediction errors, showing an overall downward trend in the plot despite the outlier STAEFormer which overfits heavily. In contrast, our model lies distinctly apart from the main trend in the lower-left corner in each plot, delivering the accuracy of models over $10\times$ larger while using dramatically fewer parameters.

Further experiments are presented in Appendix G, and a comprehensive ablation study is provided in Appendix H. We further validate the *pattern-level* interpretability of the learned undirected graph through eigenvector centrality and traffic pattern analysis in Appendix J.

6 Conclusion

To capture complex node-to-node relations in spatial-temporal data, we unroll a mixed-graph optimization algorithm into a lightweight transformer-like neural net, where an undirected graph models spatial correlations and a directed graph models temporal relationships. We show that the two graph learning modules play the role of self-attention, meaning that our unrolled network is a transformer. We design new ℓ_2 and ℓ_1-norm regularizers to quantify and promote signal smoothness on directed graphs. We interpret the derived processing operations from a designed ADMM algorithm as low-pass filters. Experiments demonstrate competitive prediction performance at drastically reduced parameter counts. One limitation of our approach is that our computed Mahalanobis distances $\mathrm{d}^u(i,j)$ and $\mathrm{d}^d(i,j)$ for respective undirected and directed graphs are non-negative (due to learned PSD metric matrices $\mathbf{M}$ and $\mathbf{P}$), whereas affinity $e(i,j)$ in traditional self-attention can be negative. We will study extensions to signed distances and more complex graph modeling as future work.

References

Saghar Bagheri, Gene Cheung, Tim Eadie, and Antonio Ortega. Joint signal interpolation / time-varying graph estimation via smoothness and low-rank priors. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 9646–9650, 2024. doi: 10.1109/ICASSP48485.2024.10447459.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL `https://api.semanticscholar.org/CorpusID:11212020`.

Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Y. Bai, G. Cheung, X. Liu, and W. Gao. Graph-based blind image deblurring from a single photograph. *IEEE Transactions on Image Processing*, 28, no.3:1404–1418, 2019.

Peter Battaglia, Jessica Blake Chandler Hamrick, Victor Bapst, Alvaro Sanchez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andy Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Jayne Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv*, 2018. URL https://arxiv.org/pdf/1806.01261.pdf.

Phillip Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92(5): 1170–1182, 1987. ISSN 00029602, 15375390. URL http://www.jstor.org/stable/2780000.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge, 2004.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. In *Foundations and Trends in Optimization*, volume 3, no.1, pp. 1–122, 2011.

Alexandru Brateanu, Raul Balmez, Adrian Avram, Ciprian Orhei, and Cosmin Ancuti. Lyt-net: Lightweight yuv transformer-based network for low-light image enhancement. *IEEE Signal Processing Letters*, 32:2065–2069, 2025. doi: 10.1109/LSP.2025.3563125.

Changlu Chen, Yanbin Liu, Ling Chen, and Chengqi Zhang. Bidirectional spatial-temporal adaptive transformer for urban traffic flow forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 34(10):6913–6925, 2022.

Hui Chen, Viet Luong, Lopamudra Mukherjee, and Vikas Singh. SimpleTM: A simple baseline for multivariate time series forecasting. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=oANkBaVci5.

S. Chen, A. Sandryhaila, J. Moura, and J. Kovacevic. Signal recovery on graphs: Variation minimization. In *IEEE Transactions on Signal Processing*, volume 63, no.17, pp. 4609–4624, September 2015.

Yang Chen, Cheng Cheng, and Qiyu Sun. Graph fourier transform based on singular value decomposition of the directed laplacian. *Sampling Theory, Signal Processing, and Data Analysis*, 21 (2):1–19, 2023. doi: 10.1007/s43670-023-00062-w. URL https://doi.org/10.1007/s43670-023-00062-w.

G. Cheung, E. Magli, Y. Tanaka, and M. Ng. Graph spectral image processing. In *Proceedings of the IEEE*, volume 106, no.5, pp. 907–930, May 2018.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs, 2021. URL https://arxiv.org/abs/2012.09699.

Aref Einizade, Fragkiskos Malliaros, and Jhony H Giraldo. Continuous product graph neural networks. *Advances in Neural Information Processing Systems*, 37:90226–90252, 2024.

Yuchen Fang, Yanjun Qin, Haiyong Luo, Fang Zhao, Bingbing Xu, Liang Zeng, and Chenxing Wang. When spatio-temporal meet wavelets: Disentangled traffic forecasting via efficient spectral graph attention networks. In *2023 IEEE 39th international conference on data engineering (ICDE)*, pp. 517–529. IEEE, 2023.

Aosong Feng and Leandros Tassiulas. Adaptive graph spatial-temporal transformer network for traffic forecasting. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, pp. 3933–3937, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392365. doi: 10.1145/3511808.3557540. URL https://doi.org/10.1145/3511808.3557540.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

Haotian Gao, Renhe Jiang, Zheng Dong, Jinliang Deng, Yuxin Ma, and Xuan Song. Spatial-temporal-decoupled masked pre-training for spatiotemporal forecasting. In *Proceedings of the Thirty-ThirdInternational Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2024.

Zhihan Gao, Xingjian Shi, Hao Wang, Yi Zhu, Yuyang (Bernie) Wang, Mu Li, and Dit-Yan Yeung. Earthformer: Exploring space-time transformers for earth system forecasting. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 25390–25403. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/a2affd71d15e8fedffe18d0219f4837a-Paper-Conference.pdf.

Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 34(11):5415–5428, 2022. doi: 10.1109/TKDE.2021.3056502.

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

Thomas Hawkins. Cauchy and the spectral theory of matrices. *Historia Mathematica*, 2(1):1–29, 1975. ISSN 0315-0860. doi: https://doi.org/10.1016/0315-0860(75)90032-4. URL https://www.sciencedirect.com/science/article/pii/0315086075900324.

Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2 edition, 2012.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Guangyu Huo, Yong Zhang, Boyue Wang, Junbin Gao, Yongli Hu, and Baocai Yin. Hierarchical spatio–temporal graph convolutional networks and transformer network for traffic flow forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 24(4):3855–3867, 2023.

Jiawei Jiang, Chengkai Han, Wayne Xin Zhao, and Jingyuan Wang. Pdformer: Propagation delay-aware dynamic long-range transformer for traffic flow prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 4365–4373, 2023.

Qizhao Jin, Xinbang Zhang, Xinyu Xiao, Ying Wang, Shiming Xiang, and Chunhong Pan. Preformer: Simple and efficient design for precipitation nowcasting with transformers. *IEEE Geoscience and Remote Sensing Letters*, 21:1–5, 2024. doi: 10.1109/LGRS.2023.3325628.

Vassilis Kalofolias, Andreas Loukas, Dorina Thanou, and Pascal Frossard. Learning time varying graphs. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2826–2830, 2017. doi: 10.1109/ICASSP.2017.7952672.

Semin Kwak, Laura Shimabukuro, and Antonio Ortega. Frequency analysis and filter design for directed graphs with polar decomposition. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 9661–9665, 2024. doi: 10.1109/ICASSP48485.2024.10446657.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

Yuejiang Li, H. Vicky Zhao, and Gene Cheung. Eigen-decomposition-free directed graph sampling via Gershgorin disc alignment. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10096903.

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*, pp. 4125–4129, 2023.

Hou-I Liu, Marco Galindo, Hongxia Xie, Lai-Kuan Wong, Hong-Han Shuai, Yung-Hui Li, and Wen-Huang Cheng. Lightweight deep learning for resource-constrained environments: A survey. *ACM Comput. Surv.*, 56(10), June 2024. ISSN 0360-0300. doi: 10.1145/3657282. URL https://doi.org/10.1145/3657282.

Antonio G. Marques, Santiago Segarra, and Gonzalo Mateos. Signal processing on directed graphs: The role of edge directionality when processing and learning from network data. *IEEE Signal Processing Magazine*, 37(6):99–116, 2020. doi: 10.1109/MSP.2020.3014597.

Rahul Mishra and Hari Prabhat Gupta. Designing and training of lightweight neural networks on edge devices using early halting in knowledge distillation. *IEEE Transactions on Mobile Computing*, 23 (5):4665–4677, 2024. doi: 10.1109/TMC.2023.3297026.

Vishal Monga, Yuelong Li, and Yonina C. Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, 2021. doi: 10.1109/MSP.2020.3016905.

Koen Oostermeijer, Qing Wang, and Jun Du. Lightweight causal transformer with local self-attention for real-time speech enhancement. In *Proceedings of Interspeech 2021*, pp. 1614–1618, Brno, Czechia, August 2021. ISCA. URL https://www.isca-archive.org/interspeech_2021/oostermeijer21_interspeech.pdf.

A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura, and P. Vandergheynst. Graph signal processing: Overview, challenges, and applications. In *Proceedings of the IEEE*, volume 106, no.5, pp. 808–828, May 2018.

Ishak Pacal. Maxcervixt: A novel lightweight vision transformer-based approach for precise cervical cancer detection. *Knowledge-Based Systems*, 289:111482, 2024. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2024.111482. URL https://www.sciencedirect.com/science/article/pii/S0950705124001175.

J. Pang and G. Cheung. Graph Laplacian regularization for inverse imaging: Analysis in the continuous domain. In *IEEE Transactions on Image Processing*, volume 26, no.4, pp. 1770–1785, April 2017.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*, 7(1):5, 2017.

Raksha Ramakrishna, Hoi-To Wai, and Anna Scaglione. A user guide to low-pass graph signal processing and its applications: Tools and applications. *IEEE Signal Processing Magazine*, 37(6): 74–85, 2020. doi: 10.1109/MSP.2020.3014590.

Gregory C Reinsel. *Elements of multivariate time series analysis*. Springer Science & Business Media, 2003.

Luana Ruiz, Fernando Gama, and Alejandro Ribeiro. Gated graph recurrent neural networks. *IEEE Transactions on Signal Processing*, 68:6303–6318, 2020.

Aliaksei Sandryhaila and José M. F. Moura. Discrete signal processing on graphs: Frequency analysis. *IEEE Transactions on Signal Processing*, 62(12):3042–3054, 2014. doi: 10.1109/TSP.2014.2321121.

Bastian Seifert and Markus Püschel. Digraph signal processing with generalized boundary conditions. *IEEE Transactions on Signal Processing*, 69:1422–1437, 2021. doi: 10.1109/TSP.2021.3051267.

Rasoul Shafipour, Ali Khodabakhsh, Gonzalo Mateos, and Evdokia Nikolova. A directed graph fourier transform with spread frequency components. *IEEE Transactions on Signal Processing*, 67 (4):946–960, 2019. doi: 10.1109/TSP.2018.2886151.

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 & Knowledge Management*, pp. 4454–4458, 2022.

J. R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, USA, 1994.

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.

Le Sun, Xinyu Wang, Yuhui Zheng, Zebin Wu, and Liyong Fu. Multiscale 3-d–2-d mixed cnn and lightweight attention-free transformer for hyperspectral and lidar classification. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–16, 2024. doi: 10.1109/TGRS.2024.3367374.

Xuxiang Ta, Zihan Liu, Xiao Hu, Le Yu, Leilei Sun, and Bowen Du. Adaptive spatio-temporal graph neural network for traffic forecasting. *Knowledge-based systems*, 242:108199, 2022.

Tam Thuc, Parham Eftekhar, Seyed Alireza Hosseini, Gene Cheung, and Philip A. Chou. Interpretable lightweight transformer via unrolling of learned graph smoothness priors. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 6393–6416. Curran Associates, Inc., 2024.

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

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416, 2018. URL http://arxiv.org/abs/1803.07416.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks, 2018. URL https://arxiv.org/abs/1710.10903.

Martin Vetterli and Jelena Kovacevic. Wavelets and subband coding. In *Prentice Hall Signal Processing Series*, 2013. URL https://api.semanticscholar.org/CorpusID:10506924.

Michał Woźniak, Bartosz Zawadzki, Michał Słowik, and Tomasz Kajdanowicz. Overview of the transformer-based models for nlp tasks. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*, pp. 407–414. IEEE, 2020. ISBN 978-83-955416-7-4. doi: 10.15439/2020F20. URL https://ieeexplore.ieee.org/document/9222960.

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*, IJCAI'19, pp. 1907–1913. AAAI Press, 2019. ISBN 9780999241141.

Mingxing Xu, Wenrui Dai, Chunmiao Liu, Xing Gao, Weiyao Lin, Guo-Jun Qi, and Hongkai Xiong. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908*, 2020.

Koki Yamada, Yuichi Tanaka, and Antonio Ortega. Time-varying graph learning based on sparseness of temporal variation. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5411–5415, 2019. doi: 10.1109/ICASSP.2019.8682762.

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-2018, pp. 3634–3640. International Joint Conferences on Artificial Intelligence Organization, July 2018. doi: 10.24963/ijcai.2018/505. URL http://dx.doi.org/10.24963/ijcai.2018/505.

Yaodong Yu, Sam Buchanan, Druv Pai, Tianzhe Chu, Ziyang Wu, Shengbang Tong, Benjamin Haeffele, and Yi Ma. White-box transformers via sparse rate reduction. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 9422–9457. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1e118ba9ee76c20df728b42a35fb4704-Paper-Conference.pdf.

Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5728–5739, 2022.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? 2023.

Haixin Zhao and Nilesh Madhu. Study of lightweight transformer architectures for single-channel speech enhancement. In *Proceedings of the 33rd European Signal Processing Conference (EUSIPCO)*, Isola delle Femmine - Palermo, Italy, September 2025. EURASIP. URL https://arxiv.org/abs/2505.21057. Accepted for publication.

Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 1234–1241, 2020.

Hui Zou and Trevor Hastie. Regularization and Variable Selection Via the Elastic Net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320, 03 2005. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2005.00503.x. URL https://doi.org/10.1111/j.1467-9868.2005.00503.x.

A Related Works

A.1 Lightweight Learning Models

Given the importance of parameter reduction for increasingly large deep learning models, there exist various studies towards this common goal across application domains. For large language models (LLMs), smaller transformer models have been achieved via low-rank assumption (Hu et al., 2022) or parameter quantization (Frantar et al., 2023). For speech enhancement, Oostermeijer et al. (2021) proposes local causal self-attention, while Zhao & Madhu (2025) proposes parameter sharing across spectral and time dimensions. For low-light image enhancement, Brateanu et al. (2025) processes the Y-channel via convolution, pooling, and multi-head self-attention, separately from the U- and V-channels. For hyperspectral image classification, Sun et al. (2024) proposes a novel multiscale 3D-2D mixed CNN design. For cervical cancer detection, Pacal (2024) substitutes larger MBConv blocks in the MaxViT architecture with smaller ConvNeXtv2 blocks, and MLP blocks with GRN-based MLPs. Similar architecture miniaturization techniques have been proposed for edge devices (Mishra & Gupta, 2024) and resource-constrained environments (Liu et al., 2024).

For traffic prediction, there are practical use cases where parameter reduction is important. For example, drivers in a city predict local traffic conditions (in an area-specific neighborhood graph) on their memory-constrained mobile devices every five minutes to compute optimal routes to their destinations, based on updated congestion information shared in their local neighborhoods, instead of sending individual requests to the cloud and overwhelming the central server.

Thuc et al. (2024) achieves a lightweight model for image interpolation in a more principled manner: first design a linear-time optimization algorithm minimizing a chosen *undirected* graph smoothness prior (such as *graph Laplacian regularizer* (GLR) (Pang & Cheung, 2017) or *graph total variation* (GTV) (Bai et al., 2019)), then unroll its iterations into neural layers to compose a "white-box" transformer-like neural net for data-driven parameter learning. This algorithm unrolling approach results in a lightweight *and* mathematically interpretable network. Our current work differs from Thuc et al. (2024) in that we model the more complex node-to-node relationships in spatial-temporal data using a *mixed* graph: i) an *undirected* graph $\mathcal{G}^u$ to capture spatial correlations among geographically

near stations, and ii) a *directed* graph $\mathcal{G}^d$ to capture sequential relationships along the temporal dimension. In particular, the notion of "frequencies" for directed graphs is not well understood in the *graph signal processing* (GSP) community. We are the first to define this notion and promote low-pass signal reconstruction via our designed ℓ_2-norm *directed graph Laplacian regularizer* (DGLR) variational term in eq. (2) and ℓ_1-norm *directed graph total variation* (DGTV) variational term in eq. (3), both derived from the *graph shift operator* (GSO) (Chen et al., 2015). To the best of our knowledge, we are the first define and promote smooth signals on a directed graph in a data-driven transformer setting.

A.2 Deep Models for Traffic Forecasting

Traffic forecasting problems contain spatial and temporal correlations between stations and time steps; thus, Graph Neural Network (GNN) is naturally employed in early research. Early research, such as STGCN (Yu et al., 2018), STSGCN (Song et al., 2020) and GGRNN (Ruiz et al., 2020), employs Graph Convolutional Networks (GCNs) to capture spatial relations and uses 1D convolutions and RNNs to process temporal relationships. ASTGNN (Ta et al., 2022), Graph WaveNet (Wu et al., 2019) and AGCRN (Bai et al., 2019) explored adaptive graph settings to capture potential non-local signal relations over the graph, trading in interpretability for improved performance. Graph Attention Networks (GATs) learn graph weights from embedded signals to enable flexible and data-targeted graph learning, typical examples include GMAN (Zheng et al., 2020) and ST-Wave (Fang et al., 2023). All GNNs above focus only on the spatial correlations, while the product graph (PG) is an intuitive model that describes local spatial-temporal relations at the same time. Einizade et al. (2024) presents a continuous product graph neural network for spatial-temporal forecasting, which achieves superior performance to the previous most-popular graph methods.

Transformers extend the powerful self-attention mechanism to all spatial-temporal data points, which enhances model performance with an extra-large number of parameters. Popular transformer-based models for traffic forecasting employ two transformers to process spatial and temporal dimensions separately (Xu et al., 2020; Chen et al., 2022; Huo et al., 2023), while state-of-the-art transformers introduce adaptive local attentions (Liu et al., 2023), time-delay (Jiang et al., 2023), or pre-training (Gao et al., 2024).

In recent years, simple linear models have been proposed to challenge the effectiveness and efficiency of heavy transformers. DLinear (Zeng et al., 2023) simply separates the input signals as trends and residuals, and employs 2 simple linear layers shared across nodes for prediction, but beats some of the transformer baselines in *long-term* forecasting. Other representative simple linear models include STID (Shao et al., 2022) and SimpleTM (Chen et al., 2025).

As shown in Table 1, we select two most representative and well-performing models from each category as baselines for comparison.

A.3 Studies on Directed Graph Laplacians

In the DGSP literature, frequencies of a general graph (directed or undirected) were first defined in Sandryhaila & Moura (2014) via Jordan decomposition of the adjacency matrix, which is known to be numerically unstable. In Shafipour et al. (2019), a set of orthornormal vectors (deemed directed graph frequencies) are computed via a minimization of spectral dispersion in a constrained quadratic programming formulation in eq. (8), which is computationally expensive. In Marques et al. (2020), a graph shift operator (GSO) $\mathbf{S}$ is assumed to be diagonalizable into $\mathbf{S} = \mathbf{V}\text{diag}(\{\lambda_k\})\mathbf{V}^{-1}$, where the generally non-orthogonal eigenvectors in $\mathbf{V}$ are deemed frequency modes and ordered using a total variation measure in eq. (2), given eigenvalues λ_k's can be complex-valued. However, asymmetric adjacency and graph Laplacian matrices are not always diagonalizable via the Spectral Theorem. Seifert & Püschel (2021) proposes to augment a directed graph with extra edges to destroy nontrivial Jordan blocks, so that the resulting adjacency matrix becomes diagonalizable. Adding edges, however, fundamentally changes the graph representation of the original pairwise relationships. Chen et al. (2023) proposes to use singular value decomposition (SVD) on the directed Laplacian to derive frequencies for directed graphs, which is computationally expensive. Kwak et al. (2024) proposes to decompose the asymmetric Laplacian matrix into a unitary part (capturing rotation or

cycles) and a positive semi-definite part (capturing diffusion-like behavior) also using SVD and polar decomposition, thus suffering the same computation burden. None of these related works proposed variational terms to define directed graph frequencies, which in turn can be simply added to an objective to promote signal smoothness during optimization without first computing frequency components, as we have done.

B Proof of Theorem 3.1

We prove Theorem 3.1 by construction as follows. A directed line graph $\mathcal{G}^d$ of N nodes with inter-node edge weights 1 (and self-loop of weight 1 at node 1) has adjacency matrix $\mathbf{W}_r^d$ and random-walk Laplacian matrix $\mathbf{L}_r^d = \mathbf{I}_N - \mathbf{W}_r^d$ of the following forms:

$$\mathbf{W}_r^d = \begin{bmatrix} 1 & 0 & \dots & & 0 \\ 1 & 0 & \dots & & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & & 0 \\ 0 & \dots & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{L}_r^d = \begin{bmatrix} 0 & 0 & \dots & & 0 \\ -1 & 1 & \dots & & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & & 0 \\ 0 & \dots & 0 & -1 & 1 \end{bmatrix}. \tag{24}$$

Note that rows 2 to N of $\mathbf{L}_r^d$ actually compose the *incidence* matrix of an undirected line graph $\mathcal{G}^u$ with edge weights 1. Thus, the symmetrized directed graph Laplacian $\mathcal{L}_r^d = (\mathbf{L}_r^d)^\top \mathbf{L}_r^d$ is also the combinatorial graph Laplacian $\mathbf{L}^u$:

$$\mathcal{L}_r^d = \mathbf{L}^u = \begin{bmatrix} 1 & -1 & 0 & \dots & & 0 \\ -1 & 2 & -1 & 0 & \dots & 0 \\ 0 & -1 & 2 & -1 & 0 \dots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \\ 0 & \dots & 0 & -1 & 2 & -1 \\ 0 & \dots & & 0 & -1 & 1 \end{bmatrix}. \tag{25}$$

As an illustration, consider a 4-node directed line graph $\mathcal{G}^d$ with edge weights 1; see Fig. 2(a) for an illustration. We see that the symmetrized directed graph Laplacian $\mathcal{L}_r^d$ defaults to the graph Laplacian $\mathbf{L}^u$ for a 4-node *undirected* line graph $\mathcal{G}^u$ (Fig. 2(b)):

$$\mathbf{W}_r^d = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \mathbf{L}_r^d = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}, \mathcal{L}_r^d = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \tag{26}$$

C Interpreting Linear System eq. (10)

For notation simplicity, we first define

$$\mathbf{r}^\tau \triangleq (\mathbf{L}_r^d)^\top \left(\frac{\boldsymbol{\gamma}^\tau}{2} + \frac{\rho}{2}\boldsymbol{\phi}^\tau\right) - \frac{\boldsymbol{\gamma}_u^\tau}{2} + \frac{\rho_u}{2}\mathbf{z}_u^\tau - \frac{\boldsymbol{\gamma}_d^\tau}{2} + \frac{\rho_d}{2}\mathbf{z}_d^\tau. \tag{27}$$

Solution $\mathbf{x}^{\tau+1}$ to linear system eq. (10) can now be written as

$$\mathbf{x}^{\tau+1} = \left(\mathbf{H}^\top\mathbf{H} + \frac{\rho}{2}\mathcal{L}_r^d + \frac{\rho_u + \rho_d}{2}\mathbf{I}\right)^{-1} (\mathbf{r}^\tau + \mathbf{H}^\top\mathbf{y}). \tag{28}$$

Define $\tilde{\mathcal{L}}_r^d \triangleq \mathbf{H}^\top\mathbf{H} + \frac{\rho}{2}\mathcal{L}_r^d$. Note that $\mathbf{H}^\top\mathbf{H}$ is a diagonal matrix with zeros and ones (corresponding to chosen sample indices) along its diagonal, and thus $\tilde{\mathcal{L}}_r^d$ is a scaled variant of $\mathcal{L}_r^d$ with the addition of selected self-loops of weight 1. Given that $\mathcal{L}_r^d = (\mathbf{L}_r^d)^\top \mathbf{L}_r^d$ is real and symmetric, $\tilde{\mathcal{L}}_r^d$ is also real and symmetric. By eigen-decomposing $\tilde{\mathcal{L}}_r^d = \tilde{\mathbf{U}}\mathrm{diag}(\{\tilde{\xi}_k\})\tilde{\mathbf{U}}^\top$, we can rewrite $\mathbf{x}^{\tau+1}$ as

$$\mathbf{x}^{\tau+1} = \tilde{\mathbf{U}}\mathrm{diag}\left(\left\{\frac{1}{\frac{\rho_u+\rho_d}{2} + \tilde{\xi}_k}\right\}\right)\tilde{\mathbf{U}}^\top\left(\mathbf{r}^\tau + \mathbf{H}^\top\mathbf{y}\right) \tag{29}$$

which demonstrates that $\mathbf{x}^{\tau+1}$ is a low-pass filter output of up-sampled $\mathbf{H}^\top\mathbf{y}$ (with bias $\mathbf{r}^\tau$).

D Deriving Solution to Optimization eq. (15)

The optimization for $\boldsymbol{\phi}$ in objective eq. (15) can be performed entry-by-entry. Specifically, for the i-th entry ϕ_i:

$$\min_{\phi_i} \; g(\phi_i) = \mu_{d,1}|\phi_i| + \gamma_i^\tau \left(\phi_i - (\mathbf{L}_r^d)_i \mathbf{x}^{\tau+1}\right) + \frac{\rho}{2}\left(\phi_i - (\mathbf{L}_r^d)_i \mathbf{x}^{\tau+1}\right)^2 \tag{30}$$

where $(\mathbf{L}_r^d)_i$ denotes the i-row of $\mathbf{L}_r^d$. $g(\phi_i)$ is convex and differentiable when $\phi_i \neq 0$. Specifically, when $\phi_i > 0$, setting the derivative of $g(\phi_i)$ w.r.t. ϕ_i to zero gives

$$\begin{aligned} \mu_{d,1} + \gamma_i^\tau + \rho\left(\phi_i^* - (\mathbf{L}_r^d)_i \mathbf{x}^{\tau+1}\right) &= 0 \\ \phi_i^* &= (\mathbf{L}_r^d)_i \mathbf{x}^{\tau+1} - \rho^{-1}\gamma_i^\tau - \rho^{-1}\mu_{d,1} \end{aligned} \tag{31}$$

which holds only if $(\mathbf{L}_r^d)_i \mathbf{x}^{\tau+1} - \rho^{-1}\gamma_i^\tau > \rho^{-1}\mu_{d,1}$. On the other hand, when $\phi_i < 0$,

$$\begin{aligned} -\mu_{d,1} + \gamma_i^\tau + \rho\left(\phi_i^* - (\mathbf{L}_r^d)_i \mathbf{x}^{\tau+1}\right) &= 0 \\ \phi_i^* &= (\mathbf{L}_r^d)_i \mathbf{x}^{\tau+1} - \rho^{-1}\gamma_i^\tau + \rho^{-1}\mu_{d,1} \end{aligned} \tag{32}$$

which holds only if $(\mathbf{L}_r^d)_i \mathbf{x}^{\tau+1} - \rho^{-1}\gamma_i^\tau < -\rho^{-1}\mu_{d,1}$. Summarizing the two cases, we get

$$\begin{aligned} \delta &= (\mathbf{L}_r^d)_i \mathbf{x}^{\tau+1} - \rho^{-1}\gamma_i^\tau \\ \phi_i^* &= \text{sign}(\delta) \cdot \max(|\delta| - \rho^{-1}\mu_{d,1},\, 0) \end{aligned} \tag{33}$$

Alternative solution with subgradients: Denote $g_1(\boldsymbol{\phi}) = \mu_{d,1}\|\boldsymbol{\phi}\|_1, g_2(\boldsymbol{\phi}) = (\boldsymbol{\gamma}^\tau)^\top(\boldsymbol{\phi} - \mathbf{L}_r^d \mathbf{x}^{\tau+1}) + \frac{\rho}{2}\|\boldsymbol{\phi} - \mathbf{L}_r^d \mathbf{x}^{\tau+1}\|_2^2$, the subgradient of $g(\boldsymbol{\phi})$ is

$$\partial g(\boldsymbol{\phi}) = \{\mathbf{g} + \nabla g_2(\boldsymbol{\phi}) | \mathbf{g} \in \partial g_1(\boldsymbol{\phi})\} = \{\mu_{d,1}\sigma + \boldsymbol{\gamma}^\tau + \rho(\boldsymbol{\phi} - \mathbf{L}_r^d \mathbf{x}^{\tau+1}) | \sigma \in \partial(\|\boldsymbol{\phi}\|_1)\}, \tag{34}$$

where

$$\partial_i(\|\boldsymbol{\phi}\|_1) = \begin{cases} \text{sign}(\phi_i), & \phi_i \neq 0, \\ [-1, 1], & \phi_i = 0. \end{cases} \tag{35}$$

The minimal $\boldsymbol{\phi}^*$ satisfies $\mathbf{0} \in \partial g(\boldsymbol{\phi}^*)$, thus

$$\begin{aligned} &\mu_{d,1} + \gamma_i^\tau + \rho(\phi_i^* - (\mathbf{L}_r^d)_i \mathbf{x}) = 0, && \text{if } \phi_i^* > 0, \\ &-\mu_{d,1} + \gamma_i^\tau + \rho(\phi_i^* - (\mathbf{L}_r^d)_i \mathbf{x}) = 0, && \text{if } \phi_i^* < 0, \\ &\exists \sigma \in [-1, 1], \text{s.t. } \mu_{d,1}\sigma + \gamma_i^\tau + \rho(\phi_i^* - (\mathbf{L}_r^d)_i \mathbf{x}) = 0, && \text{if } \phi_i^* = 0, \end{aligned} \tag{36}$$

which gives

$$\phi_i^* = \begin{cases} (\mathbf{L}_r^d)_i \mathbf{x} - \dfrac{\gamma_i^\tau}{\rho} - \dfrac{\mu_{d,1}}{\rho}, & (\mathbf{L}_r^d)_i \mathbf{x} - \dfrac{\gamma_i^\tau}{\rho} > \dfrac{\mu_{d,1}}{\rho} \\ 0, & -\dfrac{\mu_{d,1}}{\rho} \leq (\mathbf{L}_r^d)_i \mathbf{x} - \dfrac{\gamma_i^\tau}{\rho} \leq \dfrac{\mu_{d,1}}{\rho} \\ (\mathbf{L}_r^d)_i \mathbf{x} - \dfrac{\gamma_i^\tau}{\rho} + \dfrac{\mu_{d,1}}{\rho}, & (\mathbf{L}_r^d)_i \mathbf{x} - \dfrac{\gamma_i^\tau}{\rho} < -\dfrac{\mu_{d,1}}{\rho} \end{cases}. \tag{37}$$

Thus the solution of eq. (15) is

$$\boldsymbol{\phi}^{\tau+1} = \boldsymbol{\phi}^* = \text{soft}_{\frac{\mu_{d,1}}{\rho}}\left(\mathbf{L}_r^d \mathbf{x} - \frac{\gamma_i^\tau}{\rho}\right) = \text{sign}\left(\mathbf{L}_r^d \mathbf{x} - \frac{\gamma_i^\tau}{\rho}\right) \max\left(\left|\mathbf{L}_r^d \mathbf{x} - \frac{\gamma_i^\tau}{\rho}\right| - \frac{\mu_{d,1}}{\rho}, 0\right). \tag{38}$$

E Algorithms for the Unrolled Network

E.1 Parameterizing Conjugated Gradient Method in Solving Linear Equations

A linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ where $\mathbf{A}$ is a PD matrix can be solved iteratively by the conjugated gradient (CG) method, as is described in Algorithm 1. We parameterize the coefficients of the "gradients" α_k and "momentum" β_k in each CG iteration to transform the iterative algorithm to stacked neural net layers. Note that $\alpha, \beta \geq 0$ in every iteration in the original algorithm, and should be kept non-negative in the unrolled version.

Algorithm 1 Conjugate Gradient (CG) Method to Solve $\mathbf{Ax} = \mathbf{b}$ (Unrolled version $\texttt{uCG}(\mathbf{A}, \mathbf{b}, \mathbf{x}_0)$)

Require: PSD matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, vector $\mathbf{b} \in \mathbb{R}^n$, initial guess $\mathbf{x}_0$
Ensure: Approximate solution $\mathbf{x}$ such that $\|\mathbf{Ax} - \mathbf{b}\| \le \epsilon$, where ϵ is the tolerance of convergence

$\mathbf{r}_0 \leftarrow \mathbf{b} - \mathbf{Ax}_0$ ▷ *Initial residual*
$\mathbf{p}_0 \leftarrow \mathbf{r}_0$
$k \leftarrow 0$
while $\|\mathbf{Ax}_k - \mathbf{b}\| > \epsilon$ **do**
  $\boxed{\alpha_k \leftarrow \dfrac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{p}_k^\top \mathbf{A}\mathbf{p}_k}}$ ▷ *Learnable in unrolling*
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$
  $\mathbf{r}_{k+1} \leftarrow \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$ ▷ *Dual residual*
  $\boxed{\beta_k \leftarrow \dfrac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k}}$ ▷ *Learnable in unrolling*
  $\mathbf{p}_{k+1} \leftarrow \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
  $k \leftarrow k + 1$
end while
return $\mathbf{x}_{k+1}$

E.2 Detailed Algorithms for the Proposed Model

Algorithms 2 detailedly explained the operations in ADMM blocks corresponding to Figure 3(b) and (c). An overall algorithm of the unrolled network corresponding to Figure 3(a) is shown in Algorithm 3. All matrix multiplication is computed *locally* to ensure low computational and memory cost with

$$(\mathbf{L}^u\mathbf{x})_j = D^u_{j,j}x_j - \sum_{i\in\mathcal{N}_j} W_{i.j}x_j, \quad (\mathbf{L}^u_r\mathbf{x})_j = x_j - \sum_{i\in\mathcal{N}_i} \frac{W_{i,j}}{\sqrt{D^u_{i,i}D^u_{j,j}}}x_i, \tag{39}$$

$$(\mathbf{L}^d_r\mathbf{x})_j = x_j - \sum_{(i,j)\in\mathcal{E}^d} \frac{W^d_{j,i}}{D^d_{j,j}}x_i, \quad ((\mathbf{L}^d_r)^\top\mathbf{x})_j = x_j - \sum_{(j,i)\in\mathcal{E}^d} \frac{W^d_{i,j}}{D^d_{i,i}}x_i, \quad \mathcal{L}^d_r\mathbf{x} = (\mathbf{L}^d_r)^\top\mathbf{L}^d_r\mathbf{x}. \tag{40}$$

Algorithm 2 Forward Propagation of the Unrolled $\texttt{ADMM\_Block}$

Require: Initial signal output of the last ADMM block $\mathbf{x} \in \mathbb{R}^{T+S+1}$, undirected graph Laplacian matrix $\mathbf{L}^u$, directed graph Laplacian matrix $\mathbf{L}^d_r$.
Ensure: Network output of the unrolled ADMM block with M layers.

▷ ***Learnable parameters:*** *layer-wise* $\{\mu^\tau_u, \mu^\tau_{d,1}, \mu^\tau_{d,2}, \rho^\tau, \rho^\tau_u, \rho^\tau_d\}_{\tau=1}^M$, *parameter sets* $\{(\alpha_k, \beta_k)\}$ *in each* $\texttt{uCG}$ *module.*
Select real observations with mask: $\mathbf{y} \leftarrow \mathbf{Hx}, \mathbf{y} \in \mathbb{R}^{T+1}$.
for τ in $0 : M$ **do** ▷ M *ADMM layers*
  Initialize $\boldsymbol{\phi} \leftarrow \mathbf{L}^d_r\mathbf{x}$
  $\mathbf{x} \leftarrow \texttt{uCG}(\mathbf{H}^\top\mathbf{H} + \frac{\rho^\tau}{2}\mathcal{L}^d_r + \frac{\rho^\tau_u+\rho^\tau_d}{2}\mathbf{I}, (\mathbf{L}^d_r)^\top(\frac{\boldsymbol{\gamma}}{2} + \frac{\rho}{2}\boldsymbol{\phi}) - \frac{\boldsymbol{\gamma}_u}{2} + \frac{\rho^\tau_u}{2}\mathbf{z}_u - \frac{\boldsymbol{\gamma}_d}{2} + \frac{\rho^\tau_d}{2}\mathbf{z}_d + \mathbf{H}^\top\mathbf{y}, \mathbf{x})$
  $\mathbf{z}_u \leftarrow \texttt{uCG}(\mu^\tau_u\mathbf{L}^u + \frac{\rho^\tau_u}{2}\mathbf{I}, \frac{\boldsymbol{\gamma}_u}{2} + \frac{\rho^\tau_u}{2}\mathbf{x}, \mathbf{z}_u)$
  $\mathbf{z}_d \leftarrow \texttt{uCG}(\mu^\tau_{d,2}\mathcal{L}^d_r + \frac{\rho^\tau_d}{2}\mathbf{I}, \frac{\boldsymbol{\gamma}_d}{2} + \frac{\rho^\tau_d}{2}\mathbf{x}, \mathbf{z}_d)$ ▷ *Minimize* $\mathbf{x}$ *with eqs.* (10) *to* (12)
  $\boldsymbol{\phi} \leftarrow \text{soft}_{\mu^\tau_{d,1}/\rho^\tau}(\mathbf{L}^d_r\mathbf{x} - \boldsymbol{\gamma}/\rho^\tau)$ ▷ *Minimize* ϕ *with eq.* (16)
  $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} + \rho^\tau(\boldsymbol{\phi}^{\tau+1} - \mathbf{L}^d_r\mathbf{x}^{\tau+1})$
  $\boldsymbol{\gamma}_u \leftarrow \boldsymbol{\gamma}_u + \rho^\tau_u(\mathbf{x} - \mathbf{z}_u)$
  $\boldsymbol{\gamma}_d \leftarrow \boldsymbol{\gamma}_d + \rho^\tau_d(\mathbf{x} - \mathbf{z}_d)$ ▷ *Update multipliers with eqs.* (17) *and* (18)
end for
return $\mathbf{x}$

Algorithm 3 Overall Algorithm of the Proposed Unrolled Network

Require: An observation from the training set $\mathbf{y} \in \mathbb{R}^{T+1}$.
Ensure: The reconstructed signal $\mathbf{x} \in \mathbb{R}^{T+S+1}$ containing both the obseration and prediction.
▷ ***Notations:*** *Observing stations count N, time-stamp series for the entire reconstructed signal $\mathbf{t} \in \mathbb{R}^{T+S+1}$.*

$\mathbf{e}_{\text{st}} \leftarrow \texttt{STEmb}(N, \mathbf{t})$ ▷ *Shared spatial-temporal embeddings (Appendix F.3.1)*
$\mathbf{x}^0_{\text{pred}} \leftarrow \texttt{Extrapolation}(\mathbf{y}), \mathbf{x}^0_{\text{pred}} \in \mathbb{R}^S$ ▷ *Initial prediction (Appendix F.4)*
$\mathbf{x} \leftarrow [\mathbf{y}; \mathbf{x}^0_{\text{pred}}], \mathbf{x} \in \mathbb{R}^{T+S+1}$
for i in $0 : B$ **do** ▷ *B ADMM blocks and graph learning modules*
  $\mathbf{e}_i \leftarrow [\mathbf{x}_i; \mathbf{e}_{\text{st},i}], \mathbf{f}_i \leftarrow F^u(\mathbf{e}_i)$ for $i = 1, \ldots, N$ ▷ *Feature extraction (Appendix F.3.2)*
  $\mathbf{L}^{u(h)} \leftarrow \texttt{UGL}_h(\mathbf{f}_1, \ldots, \mathbf{f}_N), \quad \mathbf{L}_r^{d(h)} \leftarrow \texttt{DGL}_h(\mathbf{f}_1, \ldots, \mathbf{f}_N), h = 1, 2, \ldots, H$
  ▷ *Multi-head graph learning (Section 4.3)*
  $\mathbf{x}^{(h)}_{\text{new}} \leftarrow \texttt{ADMM\_Block}_h(\mathbf{x}, \mathbf{L}^{u(h)}, \mathbf{L}_r^{d(h)}), h = 1, 2, \ldots, H$
  $\mathbf{x}_{\text{new}} = \sum_{h=1}^{H} a_h \mathbf{x}^{(h)}_{\text{new}}$ ▷ *Learnable merge of multi-head results*
  $\mathbf{x} \leftarrow p_i \mathbf{x}_{\text{new}} + (1 - p_i)\mathbf{x}$ ▷ *Residual connections with learnable parameter p_i*
end for
return $\mathbf{x}$

F Model Setup

F.1 ADMM Blocks Setup

We initialize $\mu_u, \mu_{d,1}, \mu_{d,2}$ with 3, and the ρ, ρ_u, ρ_d with $\sqrt{N/(T+S+1)}$, where N is the number of sensors and $T+S+1$ is the full length of the recovered signal (both the observed part and the part to be predicted). Those parameters are trained every ADMM layer. We initialize the CGD parameters ($\boldsymbol{\alpha}$, $\boldsymbol{\beta}$)s as 0.08, and tune them for every CGD iteration. We clamp each element of the CGD step size $\boldsymbol{\alpha}$ to [0, 0.8] for stability. Each element of the momentum term coefficient $\boldsymbol{\beta}$ is clamped to be non-negative.

F.2 Graph Learning Modules Setup

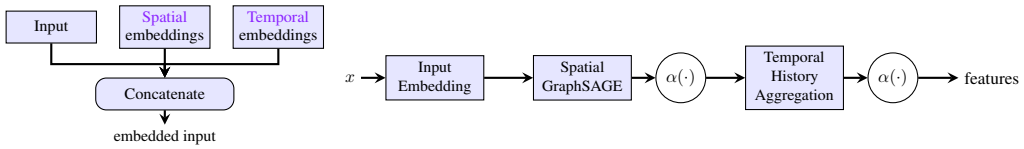
We define multiple PD matrices $\mathbf{P}$s and $\mathbf{M}$s in our unrolling model due to the different impact strengths at different times. For the undirected graph, we define $\mathbf{M}^t$ as the Mahanabolis distance matrix for the undirected graph slice at time t to vary the impacting strength between spatial neighbors in different time steps. For the directed graph, we assume that the impact strength varies with different *intervals* but remains uniform for different monitoring stations. Thus, we define $\mathbf{P}^w$ as the Mahanabolis distance matrix to represent the temporal influence of interval w for each node. Therefore, we define $(T+S+1)$ learnable PSD matrices $\mathbf{M}^0, \ldots, \mathbf{M}^{T+S}$ for undirected graph learning, and W PSD learnable matrices $\mathbf{P}^1, \ldots, \mathbf{P}^W$ for directed graph learning in total.

We learn each PSD matrix $\mathbf{M}$ in each $\texttt{UGL}$ by learning a square matrix $\mathbf{M}_0 \in \mathbb{R}^{K \times K}$ by $\mathbf{M} = \mathbf{M}_0^\top \mathbf{M}_0$, and initialize $\mathbf{M}_0$ with a matrix whose diagonal is filled with 1.5. Similarly, we learn each PSD matrix $\mathbf{P}^w$ in $\texttt{DGL}$ by learning $\mathbf{P}_0^w$ where $\mathbf{P}^w = (\mathbf{P}_0^w)^\top \mathbf{P}_0^w$. We initialize $\mathbf{P}_0^w$ with $(1 + 0.2w/W)\mathbf{I}$ to set the edge weights representing influences of longer intervals slightly smaller.

We learn 4 mixed graphs in parallel with 4 sets of matrices $\mathbf{M}^0, \ldots, \mathbf{M}^{T+S}$ and $\mathbf{P}^1, \ldots, \mathbf{P}^W$ to implement the *multi-head* attention mechanism in conventional transformers. The output signals of all 4 channels are integrated to one by a linear layer at the end of each ADMM block.

F.3 Feature Extractors Setup

We detailedly introduce our design of the feature extractor described in Section 4.3. Figure 5b shows the overall structure of our feature extractors.



(a) Input embedding layer. (b) A layer of feature extractor.

Figure 5: Framework of our feature extractor. (a) We first propose a non-parametric input embedding layer to combine signal with spatial/temporal embeddings. (b) Then aggregate embedded inputs of spatial neighbors and history information in a time window to generate a layer of feature extractor.

F.3.1 Input Embeddings

We generate *low-parametric* spatial and temporal embeddings for each node in our mixed spatio-temporal graph, see Figure 5a. For spatial embeddings, we embed each monitoring station with a 5-dimensional learnable vector, and broadcast the embeddings to every time step in the product graph. For temporal embeddings, we follow Feng & Tassiulas (2022) to use real-time-stamp, time-in-day and day-in-week embeddings. To achieve better fitting capacity while keeping the model size small, we set only the time-in-day and day-in-week embeddings *learnable*, and use *fixed* embeddings for real time stamps. We use a fixed 10-dimensional sine-cosine position embeddings (Vaswani et al., 2018) for each time stamp:

$$\begin{aligned} e[t, 2i] &= \sin(t/10000^i) \\ e[t, 2i+1] &= \cos(t/10000^i), i = 0, 1, 2, 3, 4, \end{aligned} \tag{41}$$

and set dimensions of learnable time-in-day and day-in-week embeddings as 6 and 4. All embeddings are *shared* across all graph learning modules.

We directly concatenate these spatial and temporal embedding to our signals $\mathbf{x}_j^t$ for node j at time t as $\mathbf{e}_j^t = [\mathbf{x}_j^t; \mathbf{e}_j^S; \mathbf{e}_t^T]$ for feature extractors, where $\mathbf{e}_j^S$ denote the spatial Laplacian embeddings of node j of the physical graph, and $\mathbf{e}_t^T$ denotes the temporal position embeddings of time step t.

F.3.2 Feature Extraction from Embeddings

We use a variant of GraphSAGE (Hamilton et al., 2017) to aggregate spatial features and introduce Temporal History Aggregation (THA) to aggregate spatial features. The GraphSAGE module aggregates the inputs of each node's k-NNs with a simple and uniformed linear layer to generate K-dimension spatial features for each node of each time step. The THA module aggregates the embedded inputs of past W time steps for each node with a uniformed linear layer, where W is the size of the time window. For both GraphSAGE and THA modules, we pad zeros to inputs as placeholders, and use Swish (Ramachandran et al., 2017) as acitvation functions with $\beta = 0.8$.

F.4 Constructing Graphs $\mathbf{W}^{u,0}, \mathbf{W}^{d,0}$ for the First ADMM Block

Our ADMM block recovers the full signal $\mathbf{x}$ from observed signal $\mathbf{y}$ with mixed graph $\mathcal{G}^u, \mathcal{G}^d$ constructed from embedded *full* signal $\mathbf{e} = [\mathbf{x}; \mathbf{e}^S; \mathbf{e}^T]$. Thus, for the first block in ADMM, we need an *initial extrapolation* for the signals to predict.

We propose a simple module to make an initial guess of the signals: we modify the feature extractor described in Appendix F.3.2 to generate intermediate features from the embedded input of the observed part. Then we combine the feature and temporal dimensions, and use a simple linear layer with Swish activation to generate the initially extrapolated signal.

Note: Both the feature extractor and the initial prediction module are carefully designed to involve as few parameters as possible to make the entire model lightweight. Furthermore, low-parameter and simple feature engineering ensures that the effectiveness of our model is attributed to the unrolled blocks and graph learning modules, rather than to the trivial feature engineering parts.

F.5 Training Setup

We preprocess the dataset by *standardizing* the entire time series for each node in space, but minimize a loss between the real groundtruth signal and the *rescaled* whole output signal. We use an Adam optimizer with learning rate 5×10^{-4}, together with a scheduler of reduction on validation loss with a reduction rate of 0.2 and patience of 5. We set different batch sizes for different datasets to leverage the most of the GPU memory (in this case, we use an entire Nvidia GeForce RTX 3090). Detailed selection of parameters such as number of nearest spatial neighbors k and time window size W is shown in Table 3.

Table 3: Training setup of graph construction and batch size for all traffic datasets.

Dataset	PeMS03	PeMS08	METR-LA
k	4	6	6
W	6	6	6
Batch size	12	16	16

G Further Experiments

G.1 Additional Experimental Results

Table 4 shows the performance comparison between our model and baselines in 30-/60-minute forecast on the PEMS08 (Guo et al., 2022) dataset which contains 170 nodes and 295 edges. Results show that our model also ranks top-3 in at least one metric of all experiments, and even gives the smallest MAPE among all models

Table 4: Experimental results on additional dataset PEMS08 for 30- and 60-minute forecast.The indication of bolded, highlighted and underlined entries is the same as Table 2.

Horizons	30 minutes (6 steps)			60 minutes (12 steps)		
Model	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)
VAR	26.81	17.74	12.69	28.59	18.99	13.52
STGCN	36.58	24.39	16.83	50.61	33.01	20.57
STSGCN	26.73	17.43	11.48	30.22	19.74	12.90
GMAN	**24.84**	**15.73**	10.87	**26.22**	**16.83**	13.64
ST-Wave	26.85	17.19	11.08	**27.28**	**16.85**	**11.77**
PDFormer	**22.92**	**13.91**	**9.17**	**27.36**	17.61	12.90
STAEFormer	26.54	16.62	10.88	32.68	20.78	14.86
GWN	30.95	21.65	13.56	40.24	29.12	17.79
AGCRN	34.31	19.91	10.88	38.09	22.78	12.75
SITD	27.40	18.11	12.10	33.12	22.66	20.08
SimpleTM	**24.44**	**15.07**	**9.09**	32.48	20.26	**12.34**
Ours	25.92	16.22	**10.17**	27.78	**17.22**	**11.11**

G.2 Experiments on the Entire Datasets

To further strengthen our proposed method, we ran the 60-minute forecast experiment on the PEMS03 dataset without sampling with 70 epochs of training, and the results are shown in Table 5. Our unrolled

Table 5: Performance comparison of our model to the baselines on a 60-minute forecast for the entire PEMS03 dataset. The indication of bolded, highlighted and underlined entries is the same as Table 2.

Dataset	PEMS03			METR-LA		
Model	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)
VAR	30.53	18.30	19.70	12.57	7.05	13.48
STGCN	36.77	21.81	19.37	12.27	5.88	11.79
STSGCN	28.17	17.46	16.86	12.73	5.11	10.66
PDFormer	**25.05**	**14.78**	**15.53**	12.32	**4.77**	**10.20**
GMAN	28.44	17.04	22.82	13.75	6.44	12.72
GWN	28.68	17.08	16.49	13.82	6.41	11.92
AGCRN	27.65	15.77	**15.02**	**12.18**	**4.74**	10.50
STID	26.80	**15.14**	16.45	**12.00**	**4.63**	**10.14**
SimpleTM	**24.51**	**15.32**	**15.55**	12.86	5.67	**9.89**
Ours	**26.71**	16.36	17.18	**12.22**	5.35	11.87

network still ranks top-3 in at least one of the metrics for each task, indicating the consistency of our performance with subsampling.

G.3 Inference-time Computational and Memory Cost

We compute the inference-time computational and memory cost on 60-minute forecasting for the PEMS03 dataset. We calculate the total floating-point operations (FLOPs) during inference as the computational cost with a batch size of 1, and evaluate the peak allocated memory by CUDA for each batch of size 32 as the memory cost. Results in Table 6 show that our unrolled network has the least computation cost among all the baselines except AGCRN and STID, which are generally consistently outperformed by our model. Our model's inference-time memory cost is only larger than SimpleTM and the two baselines mentioned above. In conclusion, our lightweight transformer reduces the computational cost to only 4.9% of transformer-based PDFormer and 11.2% of the GAT-based GMAN with reduced or at least comparable memory cost.

Table 6: Comparison of computational cost (GFLOPs) for each data point and inference memory cost (GB) for a batch of size 32 in 60-minute forecasting for the PEMS03 dataset.

Model*	**Ours**	STGCN	GMAN	ST-Wave	PDFormer
Computation (GFLOPS)	0.087	0.013 $\times 12^{\dagger}$	0.777	4.496	1.771
Memory (GB)	1.154	1.065$\times 12$	3.382	1.526	1.910
Model	**STAEFormer**	**GWN**	**AGCRN**	**STID**	**SimpleTM**
Computation (GFLOPS)	4.428	0.300$\times 12$	0.0002	0.036	0.670
Memory (GB)	1.846	0.411$\times 12$	0.523	0.034	0.430

* The computational and memory cost for STSGCN is not included due to the limitation of applying tools such as `thop` in the public code.
† The $\times 12$ notation in the table means that the model predicts the outputs recursively.

G.4 Training Performance w.r.t. Training Data Cost

Due to the parameter efficiency of our model, our unrolling model is also data-efficient under the same observation length. We trained our models and baselines with reduced training datasets by selecting sample windows from the sequence with larger strides of 5 and 10 in the PEMS03 dataset for 60-minute forecasting. The results below show that baseline models with large parameter counts suffer from steep performance degradation when the training data becomes scarce. In contrast, our model maintains stable performance under limited data conditions, demonstrating both lower data requirements and better generalizability. This makes it particularly suitable for deployment in data-scarce or resource-constrained environments.

Table 7: Performance under different sampling strides (metrics: MAE / RMSE / MAPE(%)) on 60-minute forecasting for PEMS03 dataset.

Model	Stride = 3 (original)	Stride = 5	Stride = 10
Ours	26.96 / 16.58 / 18.07	28.98 / 17.73 / 18.63	29.63 / 18.58 / 19.76
VAR	30.54 / 18.31 / 19.61	30.68 / 18.37 / 19.74	31.25 / 18.71 / 21.22
STSGCN	30.62 / 19.35 / 19.15	32.37 / 20.67 / 20.66	33.22 / 21.30 / 21.58
PDFormer	27.16 / 17.26 / 21.21	25.70 / 15.37 / 15.99	77.29 / 44.36 / 126.63
AGCRN	29.90 / 16.76 / 15.32	39.25 / 20.60 / 16.95	62.58 / 109.27 / 30.72

H Ablation Studies

H.1 Ablation Studies for ℓ_2- and ℓ_1- norms in Direct Graph Modeling

First, we discuss the effectiveness of the new directed graph priors, DGLR and DGTV, as well as the benefit of directed-graph modeling of temporal relations. We separately test the effects of DGLR and DGTV by removing the prior and unrolling the ADMM algorithm. We denote these models as "w/o DGTV" and "w/o DGLR". To test the benefit of the directed-graph modeling, we change the temporal graph $\mathcal{G}^d$ into a fully undirected graph $\mathcal{G}^n$ while maintaining the connections, and replace the directed graph priors with a GLR prior $\mathbf{x}^\top \mathbf{L}^n \mathbf{x}$ with the same coefficient $\mu_n = \mu_{d,2}$. We denote the undirected-time unrolling model as "UT". Detailed derivations of the ADMM algorithm for all the above ablation studies are in Appendix I.

Table 8: Performance comparison in RMSE / MAE / MAPE(%) of the ablation studies of directed-graph modeling and directed graph priors in 60-minute forecasting with PEMS03 and METR-LA.

Dataset	**Mixed-Graph**	UT	w/o DGTV	w/o DGLR
PEMS03	**26.96 / 16.58 / 18.07**	29.00 / 18.21 / 19.99	27.62 / 17.54 / 19.04	30.53 / 18.69 / 19.06
METR-LA	**12.17 / 5.27 / 11.78**	12.53 / 5.37 / 11.97	12.59 / 5.39 / 12.26	12.45 / 5.34 / 11.98

As is summarized in Table 8, removing the DGLR or DGTV term leads to noticeable drops in performance, which underscores their importance as directed graph priors. Moreover, our model outperforms the fully-undirected-graph-based model, demonstrating that sequential relationships—such as time—are better modeled as directed graphs.

H.2 Sensitivity of k, W in Graph Construction

We change the number of spatial nearest neighbors $k = 4, 6, 8$ and time window size $W = 4, 6, 8$ and present the results for PEMS03 and METR-LA dataset in Table 9. The selected hyperparameters are marked with "*". Decreasing or increasing k, W leads to performance degradation. Our selection of k, W are optimal among all test hyperparameters, considering the tradeoff with the cost of time and memory.

Table 9: Performance on 60-minute forecasting with PEMS03 (left) and METR-LA (right) under different k, W settings.

PEMS03, 60-minute forecast					METR-LA, 60-minute forecast				
k	W	RMSE	MAE	MAPE(%)	k	W	RMSE	MAE	MAPE(%)
4*	6*	**26.96**	**16.58**	**18.07**	6*	6*	**12.17**	5.27	**11.78**
4	4	27.23	16.80	17.73	4	6	12.28	5.24	11.86
4	8	27.30	17.34	19.34	8	6	12.45	5.26	11.91
6	6	29.57	18.00	19.43	6	4	12.38	**5.23**	11.84
8	6	28.40	17.99	18.48	6	8	12.52	5.42	12.29

H.3 Sensitivity of Number of Blocks and Layers in the Unrolling Model

We vary the number of blocks and ADMM layers for PEMS03 dataset and present the results in Table 10. The results show that reducing number of blocks or layers will increase the error in prediction. Our selected settings $n_{\text{blocks}} = 5, n_{\text{layers}} = 25$ is the optimal setting so far in consideration of the trade-off in computing memory and time cost.

Table 10: Performance on 60-minute forecast with PEMS03 dataset under different number of blocks and layers.

n_{blocks}	n_{layers}	PEMS03, 60-minute forecast			METR-LA, 60-minute forecast		
		RMSE	MAE	MAPE(%)	RMSE	MAE	MAPE(%)
5*	25*	**26.96**	**16.58**	18.07	**12.17**	5.27	**11.78**
5	20	27.52	17.36	18.97	12.51	5.30	11.99
5	30	27.25	17.27	18.77	12.51	5.29	11.96
4	25	27.84	17.23	**17.96**	12.43	5.32	11.93
6	25	26.97	16.92	17.97	12.46	**5.25**	11.80

H.4 Ablation Study for the Splitting-ℓ_2-term Strategy

We also ran ablations of the splitting-term strategy by optimizing $\mathbf{x}^{\tau+1}$ using only eq. (7). CG in solving eq. (7) converges much slower than in solving eq. (10)-eq. (12), and the unrolled model can become numerically unstable in training, indicating that splitting the ℓ_2 terms can stabilize the training process.

I Optimization Formulation and ADMM Algorithm in Ablation Studies

I.1 Optimization & Algorithm without DGTV Term

By removing the DGTV term $\|\mathbf{L}_r^d\mathbf{x}\|_1$ in eq. (4), and introducing the auxiliary variables $\mathbf{z}_u, \mathbf{z}_d$ as eq. (9), we rewrite the unconstrained version of the optimization by augmented Lagrangian method as

$$\min_{\mathbf{x},\mathbf{z}_u,\mathbf{z}_d} \|\mathbf{y}-\mathbf{H}\mathbf{x}\|_2^2 + \mu_u \mathbf{z}_u^\top \mathbf{L}^u \mathbf{z}_u + \mu_{d,2}\mathbf{z}_d^\top \mathcal{L}_r^d \mathbf{z}_d + (\boldsymbol{\gamma}_u^\tau)^\top(\mathbf{x}-\mathbf{z}_u) + \frac{\rho_u}{2}\|\mathbf{x}-\mathbf{z}_u\|_2^2 + (\boldsymbol{\gamma}_d^\tau)^\top(\mathbf{x}-\mathbf{z}_d) + \frac{\rho_d}{2}\|\mathbf{x}-\mathbf{z}_d\|_2^2. \tag{42}$$

We optimize $\mathbf{x}, \mathbf{z}_u$ and $\mathbf{z}_d$ in turn at iteration τ as follows:

$$\left(\mathbf{H}^\top\mathbf{H} + \frac{\rho_u+\rho_d}{2}\mathbf{I}\right)\mathbf{x}^{\tau+1} = \mathbf{H}^\top\mathbf{y} - \frac{\boldsymbol{\gamma}_u^\tau + \boldsymbol{\gamma}_d^\tau}{2} + \frac{\rho_u}{2}\mathbf{z}_u^\tau + \frac{\rho_d}{2}\mathbf{z}_d^\tau, \tag{43}$$

$$\left(\mu_u\mathbf{L}^u + \frac{\rho_u}{2}\mathbf{I}\right)\mathbf{z}_u^{\tau+1} = \frac{\boldsymbol{\gamma}_u^\tau}{2} + \frac{\rho_u}{2}\mathbf{x}^{\tau+1}, \tag{44}$$

$$\left(\mu_{d,2}\mathcal{L}_r^d + \frac{\rho_d}{2}\mathbf{I}\right)\mathbf{z}_d^{\tau+1} = \frac{\boldsymbol{\gamma}_d^\tau}{2} + \frac{\rho_d}{2}\mathbf{x}^{\tau+1}. \tag{45}$$

The updating of $\boldsymbol{\gamma}_u^\tau, \boldsymbol{\gamma}_d^\tau$ follows eq. (18).

I.2 Optimization & Algorithm without DGLR Term

By removing the DGLR term $\mathbf{x}\mathcal{L}_r^d\mathbf{x}$ in eq. (4) and introduce only the auxiliary variables $\boldsymbol{\phi}$ in eq. (5) and $\mathbf{z}_u$ in eq. (9), we rewrite the unconstrained version of the optimization of $\mathbf{x}^{\tau+1}$ by augmented Lagrangian method as

$$\min_{x,\mathbf{z}_u} \|\mathbf{y}-\mathbf{H}\mathbf{x}\|_2^2 + \mu_u\mathbf{z}_u^\top\mathbf{L}^u\mathbf{z}_u + (\boldsymbol{\gamma}^\tau)^\top(\boldsymbol{\phi}^\tau - \mathbf{L}_r^d\mathbf{x}) + \frac{\rho}{2}\|\boldsymbol{\phi}^\tau - \mathbf{L}_r^d\mathbf{x}\|_2^2 + (\boldsymbol{\gamma}_u^\tau)^\top(\mathbf{x}-\mathbf{z}_u) + \frac{\rho_u}{2}\|\mathbf{x}-\mathbf{z}_u\|_2^2 \tag{46}$$

We optimize $\mathbf{x}$ and $\mathbf{z}_u$ in turn at iteration τ as follows:

$$\left(\mathbf{H}^\top\mathbf{H}+\frac{\rho}{2}\mathcal{L}_r^d+\frac{\rho_u}{2}\mathbf{I}\right)\mathbf{x}^{\tau+1}=\mathbf{H}^\top\mathbf{y}+(\mathbf{L}_r^d)^\top(\frac{\boldsymbol{\gamma}^\tau}{2}+\frac{\rho}{2}\boldsymbol{\phi}^\tau)-\frac{\boldsymbol{\gamma}_u^\tau}{2}+\frac{\rho_u}{2}\mathbf{z}_u^\tau \tag{47}$$

$$\left(\mu_u\mathbf{L}^u+\frac{\rho_u}{2}\mathbf{I}\right)\mathbf{z}_u^{\tau+1}=\frac{\boldsymbol{\gamma}_u^\tau}{2}+\frac{\rho_u}{2}\mathbf{x}^{\tau+1} \tag{48}$$

The optimization of $\boldsymbol{\phi}^{\tau+1}$ is the same as eq. (15), and the solution is the same as eq. (16). The updating of $\boldsymbol{\gamma}, \boldsymbol{\gamma}_u$ follows eq. (17) and eq. (18).

I.3 Algorithm Unrolling with an Undirected Temporal Graph

In this section, we are changing the directed temporal graph $\mathcal{G}^d$ into an *undirected* graph while maintaining the connections. Denote the undirected temporal graph as $\mathcal{G}^n$. We introduce the GLR operator $\mathbf{x}^\top\mathbf{L}^n\mathbf{x}$ for $\mathcal{G}^n$ on signal $\mathbf{x}$ where $\mathbf{L}^n$ is the normalized Laplacian matrix of $\mathcal{G}^n$. Thus the optimization problem is as follows:

$$\min_x \|\mathbf{y}-\mathbf{H}\mathbf{x}\|_2^2+\mu_u\mathbf{x}^\top\mathbf{L}^u\mathbf{x}+\mu_n\mathbf{x}^T\mathbf{L}^n\mathbf{x} \tag{49}$$

where $\mu_u, \mu_n \in \mathbb{R}_+$ are weight parameters for the two GLR priors. The opitimization is similar to that in Appendix I.1. By a similar approach which introduces auxiliary variables $\mathbf{z}_u=\mathbf{x}, \mathbf{z}_n=\mathbf{x}$, we optimize $\mathbf{x}, \mathbf{z}_u, \mathbf{z}_n$ as follows:

$$\left(\mathbf{H}^\top\mathbf{H}+\frac{\rho_u+\rho_n}{2}\mathbf{I}\right)\mathbf{x}^{\tau+1}=\mathbf{H}^\top\mathbf{y}-\frac{\boldsymbol{\gamma}_u^\tau+\boldsymbol{\gamma}_n^\tau}{2}+\frac{\rho_u}{2}\mathbf{z}_u^\tau+\frac{\rho_n}{2}\mathbf{z}_n^\tau, \tag{50}$$

$$\left(\mu_u\mathbf{L}^u+\frac{\rho_u}{2}\mathbf{I}\right)\mathbf{z}_u^{\tau+1}=\frac{\boldsymbol{\gamma}_u^\tau}{2}+\frac{\rho_u}{2}\mathbf{x}^{\tau+1}, \tag{51}$$

$$\left(\mu_n\mathbf{L}^n+\frac{\rho_n}{2}\mathbf{I}\right)\mathbf{z}_n^{\tau+1}=\frac{\boldsymbol{\gamma}_n^\tau}{2}+\frac{\rho_n}{2}\mathbf{x}^{\tau+1}. \tag{52}$$

The updating of $\boldsymbol{\gamma}_u^\tau, \boldsymbol{\gamma}_n^\tau$ is also similar to eq. (18) as follows:

$$\boldsymbol{\gamma}_u^{\tau+1}=\boldsymbol{\gamma}_u^\tau+\rho_u(\mathbf{x}^{\tau+1}-\mathbf{z}_u^{\tau+1}), \quad \boldsymbol{\gamma}_n^{\tau+1}=\boldsymbol{\gamma}_n^\tau+\rho_n(\mathbf{x}^{\tau+1}-\mathbf{z}_n^{\tau+1}). \tag{53}$$

The undirected temporal graph $\mathcal{G}^n$ is learned by an `UGL` described in Section 4.3, and is initialized similar to the spatial UGL described in Appendix F.2. The initialization of μ_n, ρ_n and the CGD parameters in solving eq. (52) follows the setup in Appendix F.1.

J Interpreting the Learned Graph via Eigenvector Centrality

We analyze the relationship between the last undirected graph learned by the model and the observed traffic flows. For the best trained model on a 60-minute forecast on PEMS03, we calculate the learned weighted adjacency matrices $\mathbf{W}^u$ of the undirected graphs on the first 288 time steps (24 hours) in the test set. We calculate the normalized positive eigenvector corresponding to the largest eigenvalue of $\mathbf{W}^u$, known as the *Perron vector*, which is strictly positive by the Perron-Frobenius Theorem (Horn & Johnson, 2012). The Perron vector $\mathbf{v} \in \mathbb{R}^N$ reflects the *eigenvector centrality* (Bonacich, 1987) of the individual nodes: the "importance" of individual nodes, where a node is deemed important if it is connected to other important nodes, *i.e.*, $\lambda v_i = \sum_j W_{i,j}^u v_j$ for $v_j > 0$.

Figure 6 shows the evolution of the top-10 Perron vector entries and the corresponding traffic flow. First, we observe that evident disturbances in the Perron vector entries only exist when there is a rapid lift in the traffic flows from near zero to hundreds, practically corresponding to the beginning of a new morning. Furthermore, the eigenvector centrality returns to the initial state once the traffic flow becomes stable, and there are no significant changes in the ranking of the eigenvector centrality, indicating that **the node set with the largest Perron vector entries is generally consistent throughout the observation**, which marks the critical congestion hubs.

To further examine the role of the discovered important nodes in short-range interactions, we then select the four most important nodes `#328`, `#27`, `#350` and `#171` according to the elements in the Perron vectors, and plot the traffic flow of each node and the physical neighbors in selected

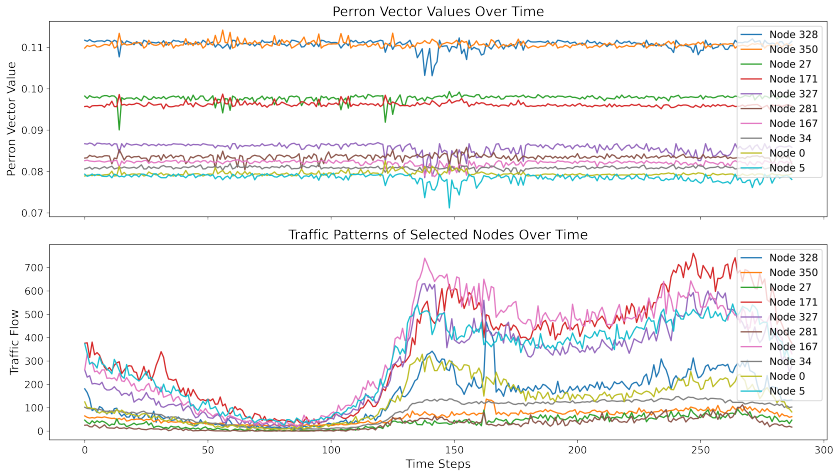
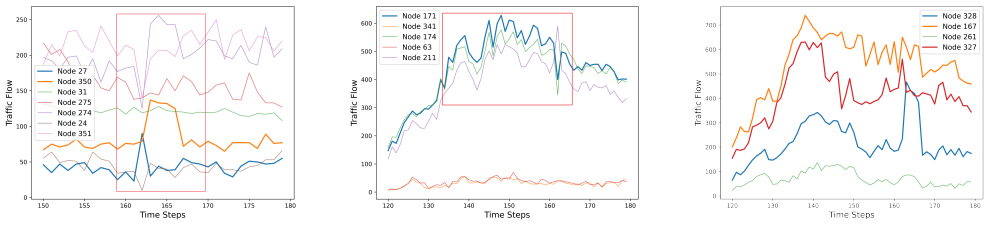


Figure 6: The evolution of eigenvector centrality in the learned undirected graph in the last block and traffic flow in the first 24 hours in the test split of the PEMS03 dataset.



(a) Neighborhood of nodes #27 and #350.

(b) Neighborhood of node #171.

(c) Neighborhood of node #328.

Figure 7: The traffic patterns of the nodes in the neighborhood of the selected important nodes by Perron vectors. The traffic flows of selected nodes are bolded.

windows of 30 and 60 steps in the daytime in Figure 7, where the selected important nodes are plotted with bolder lines (note that the neighborhood of nodes #27 and #350 are exactly the same set). In the red box in Figure 7a, the spike in node #27 results in the negative spike in the neighboring nodes #274, #24 and #351. Further, the combined effect of this spike and the bump in node #350 results in the overshoot after the downward spike in node #274. The signal trends in the red box in Figure 7b clearly reflect the time delay of the traffic flow in the neighboring nodes except the "quiet sequences" to the selected node #171. Both figures 7a and 7b demonstrate clear spatiotemporal dependency patterns captured by the learned undirected graph, where **high-centrality nodes play a dominant role in propagating local disturbances and temporal delays to their physical neighbors.** Additionally, as shown in Figure 7c, for the traffic patterns in the neighborhood of node #328, which has the largest Perron vector entry, two of its three neighboring nodes exhibit the 5th (#171) and 7th (#167) largest eigenvector centralities. This observation is consistent with the rationale that the most important node tends to be connected to other highly important nodes. Consequently, the traffic flow trends in the neighborhood of node #350 tend to be less affected by the central node, as its neighbors exhibit relatively high eigenvector centralities themselves.

In conclusion, our model successfully discovers physically interpretable graph structures from data, and the learned node importance is highly consistent with actual traffic dynamics.

Practically, our model can identify the key intersections that critically influence traffic dynamics and further guide traffic engineering in alleviating future traffic congestion.

K Error Bar for Our Model in Table 2

Table 11 shows the 2σ error bars of our model's performance in the 60-minute forecast in Table 2 in Section 5, each of which is computed from 3 runs on different random seeds.

Table 11: 2σ error bar of the performance metrics of our model in 60-minute traffic forecast in all three datasets.

Dataset	PEMS03	METR-LA	PEMS08
$2\sigma_{\text{RMSE}}$	0.05	0.04	0.04
$2\sigma_{\text{MAE}}$	0.02	0.02	0.02
$2\sigma_{\text{MAPE(\%)}}$	0.03	0.04	0.04

L Reproducibility

For reproducibility, our code is available in `https://anonymous.4open.science/r/Unrolling-GSP-STForecast-6AEE`.

M LLM Usage Declaration

We affirm that no LLMs were used in research ideation, code implementation, or writing of this paper; their use was limited to correcting grammatical errors.