STC-CPR: Lightweight Spatio-Temporal Compression with Consistency-Regularized Prototype Routing for Traffic Data Imputation

Anonymous submission

Abstract

Missing observations in large-scale traffic sensor networks can degrade downstream traffic monitoring and forecasting, especially when some sensors are entirely unobserved. Existing spatio-temporal imputation methods often rely on predefined road graphs or model dependencies in the full node–time space, resulting in sensitivity to noisy relations and high computational cost. We propose STC-CPR, a lightweight compress–route–reconstruct framework for unknown-node traffic data imputation. STC-CPR learns global dependencies in a compact spatio-temporal field and uses shared prototypes to exchange recurring traffic information among its latent components. A perturbation-based consistency objective improves the stability of prototype routing, while shallow bidirectional graph propagation provides complementary local information for node-level reconstruction. Experiments on four real-world traffic datasets show that STC-CPR consistently achieves strong imputation accuracy under different unknown-node ratios while maintaining favorable robustness and computational efficiency. Further analyses demonstrate the effectiveness of its compact representation and prototype routing.

Introduction

Traffic sensor networks provide continuous measurements of traffic speed, flow, and occupancy for intelligent transportation systems (Yu, Yin, and Zhu 2018). However, sensor failures, communication interruptions, and maintenance frequently leave these observations incomplete, undermining downstream monitoring, forecasting, and control. Recovering missing traffic signals is therefore fundamental to reliable traffic intelligence (Cini, Marisca, and Alippi 2022).

Traffic data exhibit strong temporal and spatial dependencies: congestion evolves continuously over time, while physically or functionally related roads often show correlated dynamics. Consequently, many traffic models represent sensors as a road graph and combine temporal modeling with graph-based spatial propagation (Yu, Yin, and Zhu 2018; Li et al. 2018; Wu et al. 2019, 2020; Bai et al. 2020). This paradigm has substantially advanced spatio-temporal traffic learning.

This is an anonymized submission for review purposes only. Distribution, citation, or public sharing of this manuscript is strictly prohibited. Copyright and publication details will appear in the final version if accepted.

Despite these advances, relation-centered modeling in the original node–time space remains limited for unknown-node imputation: full-resolution models process recurring dynamics across sensors and time as separate node signals. Moreover, a completely unobserved sensor provides no historical evidence of its own and must be recovered from regularities shared by the observed network. Although traffic observations are high-dimensional, their variations are often governed by a limited set of recurring spatio-temporal factors (Bahadori, Yu, and Liu 2014). As illustrated in Figure 1, similar traffic modes recur across different roads, whereas the target sensors contain no historical observations of their own. Graph-based models can encode such factors implicitly, but many node-level propagation schemes do not explicitly separate them from sensor-specific representations or preserve them as reusable patterns. These limitations motivate a model that extracts the low-dimensional structure of the traffic field and makes its recurring patterns accessible to unobserved sensors.

Based on this motivation, we propose STC-CPR, a lightweight *compress–route–reconstruct* framework for unknown-node traffic data imputation. STC-CPR first constructs a compact spatio-temporal field from incomplete observations. A learnable spatial decomposition concentrates correlated sensor behavior into a small set of latent components, while coarse temporal modeling captures dominant dependencies among reduced temporal states. Compression is therefore used not only to improve efficiency, but also to separate shared traffic factors from redundant full-resolution observations. The learned projections retain the spatial and temporal mappings required to reconstruct sensor-level outputs.

Compression alone, however, does not preserve recurring traffic states in an explicit and reusable form. STC-CPR therefore maintains a shared prototype bank as persistent latent memory. Its learnable prototype bases store recurring traffic patterns across training samples, while the write operation combines these stored patterns with current component evidence and the read operation redistributes the resulting states to each component. This many-to-few-to-many interaction provides reusable pattern-level context for sensors without historical observations. To make prototype routing robust to perturbations of the compact field, we impose symmetric agreement between prototype-routed representations from

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

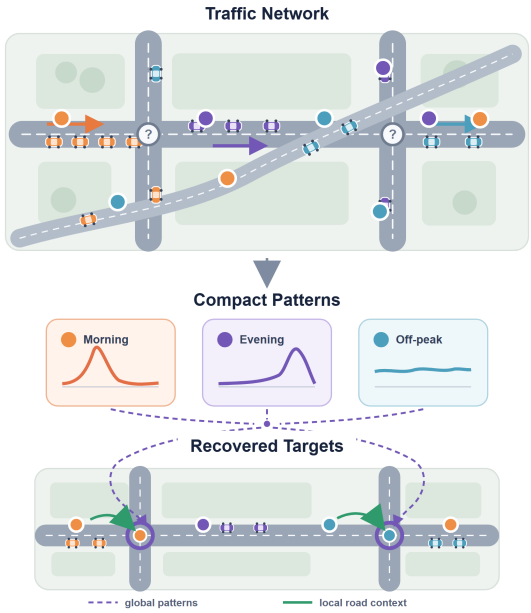


Figure 1: Traffic dynamics observed across different roads can be summarized by a small set of recurring patterns, which provide global priors for recovering unobserved sensors together with local road context.

two independently perturbed views (Chen and He 2021). This consistency constraint suppresses perturbation-specific variation and stabilizes pattern retrieval. A shallow bidirectional topology path additionally provides local road features during reconstruction.

The contributions of this work are summarized as follows:

- We propose a lightweight spatio-temporal compression mechanism that exploits the low effective dimensionality of traffic observations. It concentrates dominant shared factors in a compact field while retaining the mappings required for node- and time-level reconstruction.
- We introduce consistency-regularized prototype routing as a persistent and adaptive traffic-pattern memory. Shared prototypes store recurring latent patterns, adapt them to current observations through write–read routing, and maintain stable retrieval under compact-field perturbations.
- Extensive experiments on four real-world traffic datasets demonstrate the accuracy, robustness, and computational efficiency of STC-CPR for unknown-node traffic data imputation.

Related Work

Traffic Data Imputation

Early traffic imputation methods infer missing values from nearby or statistically related observations, including nearest-neighbor estimation (Cover and Hart 1967), inverse distance weighting (Lu and Wong 2008), and geostatistical estimation (Krige 1951; Cressie and Wikle 2011). Low-rank matrix and tensor models exploit global spatio-temporal regularity (Bahadori, Yu, and Liu 2014), while LATC further combines low-rank tensor completion with temporal autoregression for traffic imputation (Chen et al. 2022). Although interpretable, these approaches often rely on fixed assumptions or instance-specific optimization, limiting their ability to capture nonlinear traffic dynamics and adapt to changing sensor availability.

Deep learning methods learn imputation functions directly from data. BRITS exploits bidirectional recurrent dynamics (Cao et al. 2018), while SAITS uses self-attention to capture temporal dependencies and feature correlations (Du, Côté, and Liu 2023). These methods, however, mainly target temporal or entry-wise missingness rather than completely unobserved traffic sensors. PM-MemNet explores a complementary direction by matching observed traffic histories to representative patterns stored in a key–value memory for forecasting (Lee et al. 2022). Nevertheless, this forecasting-oriented memory requires an observed query sequence and does not address how representative patterns can be retrieved or adapted when the target sensor has no historical observations.

Spatio-temporal graph neural networks

Spatio-temporal graph neural networks provide a foundation for traffic modeling: STGCN combines graph and temporal convolutions, while DCRNN models directed traffic propagation through diffusion convolution and recurrent transitions (Yu, Yin, and Zhu 2018; Li et al. 2018). Graph WaveNet and MTGNN introduce adaptive graph learning for spatio-temporal dependencies (Wu et al. 2019, 2020), whereas AGCRN uses node-adaptive parameters to capture sensor-specific patterns (Bai et al. 2020).

Graph-based imputation methods extend spatio-temporal propagation to missing-data recovery. GRIN reconstructs missing observations through recurrent graph propagation (Cini, Marisca, and Alippi 2022), while KCN and IGNNK infer completely unobserved sensors from node-level relations (Appleby, Liu, and Liu 2020; Wu et al. 2021). Recent studies address more challenging settings through sparse spatio-temporal learning (Deng et al. 2024), robust kriging (Liang et al. 2026), and attention- or pattern-based modeling (Wang et al. 2025; Hu et al. 2025). Nevertheless, their dominant interaction often remains in the sensor space, motivating a compact reconstruction model that retains useful topology without relying on deep node-level propagation.

Method

Preliminaries

Let $X \in \mathbb{R}^{T\times N\times F}$ denote a complete traffic field, where T, N, and F are the numbers of time steps, sensors, and traffic features, respectively. We consider a single traffic variable, either traffic flow or speed, and therefore set $F = 1$.

The sensor set $\mathcal{V}$ is divided into observed sensors $\mathcal{V}_o$ and unknown target sensors $\mathcal{V}_u$, where $\mathcal{V}_o \cap \mathcal{V}_u = \varnothing$ and $\mathcal{V}_o \cup \mathcal{V}_u = \mathcal{V}$. Let $\tilde{X} \in \mathbb{R}^{T\times N\times 1}$ denote the incomplete traffic field, in which observations are available only for sensors in $\mathcal{V}_o$, while the complete sequences of sensors in $\mathcal{V}_u$

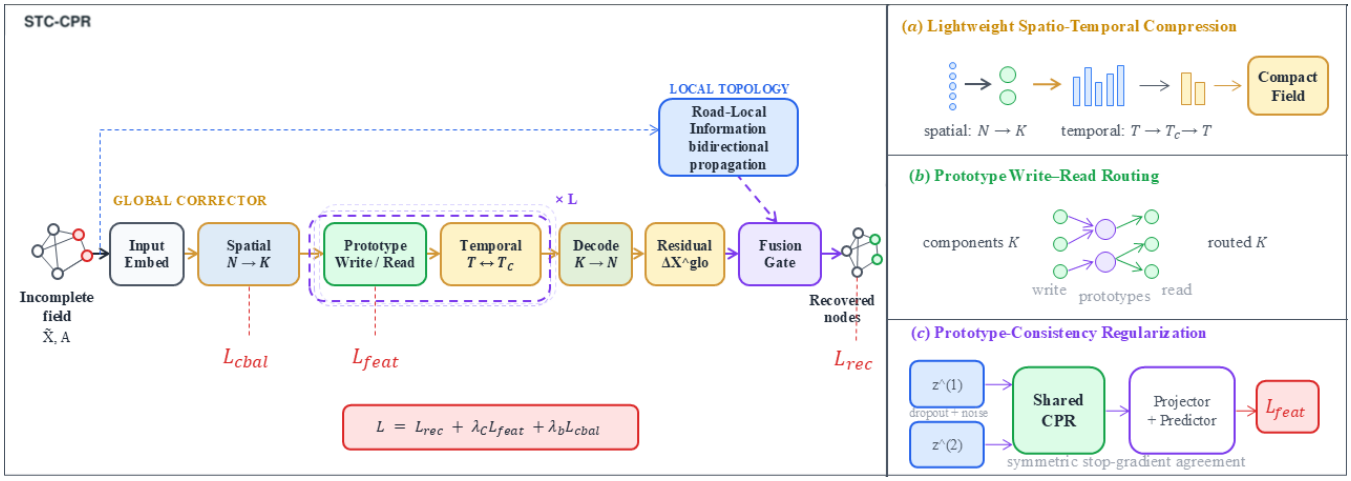


Figure 2: Overall architecture of STC-CPR. The main pathway spatially compresses the incomplete traffic field and employs stacked prototype write–read and temporal learning blocks to capture global spatio-temporal dependencies. The reconstructed global representation is combined with complementary road-local information through a fusion gate.

are unavailable. The objective of STC-CPR is to learn a reconstruction function $\mathcal{F}_\theta$, parameterized by θ, that maps the incomplete traffic field $\tilde{X}$ to

$$\hat{X} = \mathcal{F}_\theta(\tilde{X}) \in \mathbb{R}^{T\times N\times 1}. \tag{1}$$

The reconstructed sequences $\hat{X}_{:,\mathcal{V}_u,:}$ are expected to approximate the ground-truth traffic states $X_{:,\mathcal{V}_u,:}$ of the unknown sensors.

Overall Framework

As shown in Figure 2, STC-CPR learns global traffic dependencies in a compact latent space and incorporates lightweight road-local information during gated reconstruction. Prototype routing and temporal feature learning are stacked for L blocks, with the prototype bank and routing function shared across all blocks.

Lightweight Spatial Compression

The incomplete traffic field is first projected from one input channel to a d-dimensional representation $H \in \mathbb{R}^{T\times N\times d}$ using a learnable 1×1 convolution. To reduce spatial redundancy, STC-CPR represents the N sensors using K latent spatial components, where $K \ll N$. A learnable decomposition function ϕ_s generates the contribution coefficients of each sensor:

$$S_{n,:} = \mathrm{softmax}(\phi_s(H_{:,n,:})), \qquad S \in \mathbb{R}^{N\times K}, \tag{2}$$

where $S_{n,k}$ represents the contribution of sensor n to component k. The compact spatial representation is obtained through normalized aggregation:

$$H^s_{:,k,:} = \frac{\sum_{n=1}^{N} S_{n,k} H_{:,n,:}}{\sum_{n=1}^{N} S_{n,k} + \epsilon}, \qquad H^s \in \mathbb{R}^{T\times K\times d}, \tag{3}$$

where ϵ is a small constant for numerical stability.

To prevent the representation from concentrating on only a few components, we introduce a component-utilization regularizer:

$$\bar{s}_k = \frac{1}{N}\sum_{n=1}^{N} S_{n,k}, \qquad \mathcal{L}_{\mathrm{cbal}} = K\sum_{k=1}^{K}\left(\bar{s}_k - \frac{1}{K}\right)^2. \tag{4}$$

This regularization balances the aggregate utilization of the latent components without forcing each sensor to have uniform decomposition coefficients.

Prototype Write–Read Routing

Following spatial compression, the compact representation $H^s \in \mathbb{R}^{T\times K\times d}$ is refined through M_p shared prototypes $P_0 \in \mathbb{R}^{M_p\times d}$. A learnable routing function g assigns each spatial component to the prototypes:

$$R_{t,k,:} = \mathrm{softmax}\big(g(H^s_{t,k,:})\big), \qquad R \in \mathbb{R}^{T\times K\times M_p}. \tag{5}$$

During the write stage, each prototype integrates information from the spatial components according to the routing coefficients. The resulting dynamic state is combined with the corresponding learnable prototype:

$$\begin{aligned} P^{\mathrm{dyn}}_{t,m,:} &= \frac{\sum_{k=1}^{K} R_{t,k,m} H^s_{t,k,:}}{\sum_{k=1}^{K} R_{t,k,m} + \epsilon}, \\ P_{t,m,:} &= (1-\beta)P_{0,m,:} + \beta P^{\mathrm{dyn}}_{t,m,:}. \end{aligned} \tag{6}$$

where $\beta \in (0,1)$ is a learnable scalar controlling the contribution of the dynamic state.

During the read stage, the integrated prototype states are redistributed to the spatial components using the same routing coefficients:

$$H^p_{t,k,:} = \sum_{m=1}^{M_p} R_{t,k,m} P_{t,m,:}, \qquad H^p \in \mathbb{R}^{T\times K\times d}. \tag{7}$$

This write–read process enables information exchange among spatial components through a small set of shared prototypes. The prototype bottleneck also keeps the interaction lightweight. Both the write and read operations have a time complexity of $\mathcal{O}(TKM_pd)$. In comparison, explicit pairwise interaction among the K components requires $\mathcal{O}(TK^2d)$, while interaction in the original sensor space can require $\mathcal{O}(TN^2d)$.

Temporal Compression and Feature Learning

Following prototype routing, the routed representation $H^p \in \mathbb{R}^{T\times K\times d}$ is compressed along the temporal dimension. Consecutive time steps are divided into groups of size r and summarized by average pooling:

$$H^c = \mathrm{Pool}_r(H^p) \in \mathbb{R}^{T_c\times K\times d}, \qquad T_c = \frac{T}{r}, \tag{8}$$

where T_c is the number of coarse temporal states.

A temporal dependency matrix is constructed from a learnable temporal basis E_t:

$$\begin{aligned} A_t &= \mathrm{softmax}\big(\mathrm{ReLU}(E_tE_t^\top)\big) \in \mathbb{R}^{T_c\times T_c},\\ \bar{H}^c_{:,k,:} &= A_t^\top H^c_{:,k,:}. \end{aligned} \tag{9}$$

where A_t describes the dependencies among the coarse temporal states. The updated representation is projected back to the original temporal resolution using $W_t \in \mathbb{R}^{T\times T_c}$. A residual connection preserves the information obtained from prototype routing:

$$H^t_{:,k,:} = H^p_{:,k,:} + W_t\bar{H}^c_{:,k,:}, \qquad H^t \in \mathbb{R}^{T\times K\times d}. \tag{10}$$

The resulting representation H^t combines prototype-mediated spatial information with temporal dependencies learned at a reduced resolution.

Temporal dependency learning over the coarse states requires $\mathcal{O}(KT_c^2d)$ operations, compared with $\mathcal{O}(KT^2d)$ at the original temporal resolution. Since $T_c = T/r$, temporal compression reduces the corresponding interaction cost by approximately a factor of r^2.

Gated Reconstruction

The temporally refined representation $H^t \in \mathbb{R}^{T\times K\times d}$ is first projected from the K spatial components back to the N sensors using the decomposition coefficients S. A lightweight output mapping ϕ_o then produces a global reconstruction term:

$$\begin{aligned} H^n_{:,n,:} &= \sum_{k=1}^{K} S_{n,k}H^t_{:,k,:},\\ \Delta X^g &= \phi_o(H^n) \in \mathbb{R}^{T\times N\times 1}. \end{aligned} \tag{11}$$

Road topology is introduced only during reconstruction. Let $A_f \in \mathbb{R}^{N\times N}$ denote the directed road adjacency matrix and $A_b = A_f^\top$ its reverse direction. Their normalized forms $\bar{A}_f$ and $\bar{A}_b$ propagate traffic information in both directions. A shallow transformation extracts road-local information and produces a local reconstruction term:

$$\begin{aligned} H^\ell &= \phi_\ell\Big([\bar{A}_f\tilde{X} \,\|\, \bar{A}_b\tilde{X}]\Big),\\ X^\ell &= \phi_r(H^\ell) \in \mathbb{R}^{T\times N\times 1}. \end{aligned} \tag{12}$$

A fusion gate adaptively controls the contribution of the decoded global information:

$$\begin{aligned} G &= \sigma\big(\phi_g([H^\ell \,\|\, \Delta X^g])\big),\\ \hat{X} &= X^\ell + G\odot\Delta X^g, \end{aligned} \tag{13}$$

where $G \in [0,1]^{T\times N\times 1}$, $\sigma(\cdot)$ is the sigmoid function, and $\odot$ denotes element-wise multiplication. In this way, global dependencies are learned in the compact latent space, while shallow bidirectional propagation supplies complementary road-local information only during reconstruction.

The component-to-sensor projection requires $\mathcal{O}(TNKd)$ operations, while sparse bidirectional propagation requires $\mathcal{O}(T|\mathcal{E}|)$, where $|\mathcal{E}|$ is the number of road connections. The reconstruction process therefore avoids stacked graph propagation over the full sensor network.

Training Objectives

The fixed target sensors in $\mathcal{V}_u$ are excluded from both the model inputs and reconstruction supervision throughout training. To obtain supervision without exposing these targets, we randomly sample an auxiliary masked subset $\mathcal{V}_m \subset \mathcal{V}_o$ from the observed sensors and hide its measurements from the training input. The reconstruction loss is computed only on these artificially hidden sensors:

$$\mathcal{L}_{\mathrm{rec}} = \frac{1}{T|\mathcal{V}_m|}\sum_{t=1}^{T}\sum_{n\in\mathcal{V}_m}\Big(\hat{X}_{t,n,1} - X_{t,n,1}\Big)^2. \tag{14}$$

To improve the stability of prototype routing, two independently perturbed views of the spatially compressed representation are generated using feature noise and component dropout. Both views are processed by the shared prototype router $\mathcal{R}$:

$$H^{p,(1)} = \mathcal{R}(\mathcal{A}_1(H^s)), \qquad H^{p,(2)} = \mathcal{R}(\mathcal{A}_2(H^s)), \tag{15}$$

where $\mathcal{A}_1$ and $\mathcal{A}_2$ denote two independent perturbation operations. A projector h and a predictor q are then applied:

$$Y^{(i)} = h(H^{p,(i)}), \qquad Q^{(i)} = q(Y^{(i)}), \qquad i\in\{1,2\}. \tag{16}$$

The prototype-consistency loss is defined as

$$\mathcal{L}_{\mathrm{feat}} = \frac{1}{2}D\Big(Q^{(1)}, \mathrm{sg}(Y^{(2)})\Big) + \frac{1}{2}D\Big(Q^{(2)}, \mathrm{sg}(Y^{(1)})\Big), \tag{17}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient and $D(\cdot,\cdot)$ is cosine dissimilarity.

Combining the reconstruction loss, prototype-consistency loss, and component-utilization regularizer defined in the spatial compression module, the complete objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{rec}} + \lambda_c\mathcal{L}_{\mathrm{feat}} + \lambda_b\mathcal{L}_{\mathrm{cbal}}, \tag{18}$$

where λ_c and λ_b control the contributions of prototype consistency and component-utilization regularization, respectively.

Method	PEMS04			PEMS08			METR-LA			PEMS-BAY		
	MAE	RMSE	MRE	MAE	RMSE	MRE	MAE	RMSE	MRE	MAE	RMSE	MRE
30% Unknown Nodes												
KCN	77.66	102.94	36.93%	88.42	117.03	40.91%	6.89	9.79	11.77%	4.18	6.91	6.70%
IGNNK	72.79	96.08	34.69%	78.21	108.21	36.94%	6.00	8.92	10.26%	3.82	6.53	6.11%
GRIN	72.37	101.84	34.49%	85.46	111.82	39.54%	7.50	11.95	12.83%	5.11	9.51	8.17%
OPCR	73.57	104.17	36.29%	84.03	114.47	38.88%	6.06	9.58	10.37%	4.02	7.12	6.43%
DarkFarseer	76.20	106.83	36.12%	96.52	126.39	43.39%	6.35	9.38	10.86%	3.82	6.71	6.11%
STC-CPR	**68.73**	**95.78**	**31.67%**	**75.78**	**103.88**	**35.43%**	**5.67**	**8.68**	**9.70%**	**3.74**	**6.47**	**5.99%**
Std.	0.94	1.17	0.46%	0.32	0.51	0.15%	0.08	0.13	0.14%	0.17	0.26	0.16%
50% Unknown Nodes												
KCN	87.05	118.96	40.89%	91.17	117.79	38.69%	7.19	10.18	12.25%	4.12	6.76	6.61%
IGNNK	85.74	115.87	39.13%	81.47	105.59	34.57%	6.67	10.02	11.36%	4.01	6.68	6.44%
GRIN	84.63	117.86	39.76%	91.22	117.80	37.62%	8.62	12.29	14.68%	5.51	9.82	8.84%
OPCR	82.18	118.50	38.51%	84.36	113.34	36.19%	6.52	10.80	11.10%	4.05	7.17	6.50%
DarkFarseer	77.50	**108.32**	36.41%	80.12	109.42	34.58%	7.14	10.09	12.16%	4.01	6.80	6.43%
STC-CPR	**76.96**	110.38	**36.15%**	**77.57**	**103.09**	**33.25%**	**6.28**	**9.68**	**10.70%**	**3.88**	**6.44**	**6.23%**
Std.	0.44	1.29	0.18%	0.49	1.22	0.33%	0.11	0.11	0.15%	0.10	0.14	0.17%

Table 1: Comparison results on four traffic datasets. The row below each STC-CPR result reports the sample standard deviation over five runs. The best results are highlighted in bold and the second-best results are underlined.

Experiments

Experimental Setup

Datasets and protocol. We evaluate STC-CPR on four real-world traffic datasets: PEMS04, PEMS08, PEMS-BAY (Chen et al. 2001), and METR-LA (Jagadish et al. 2014). The four datasets contain 307, 170, 325, and 207 sensors, respectively, and are sampled every 5 minutes. We impute traffic flow on PEMS04 and PEMS08 and traffic speed on METR-LA and PEMS-BAY. For each dataset-ratio setting, 30% or 50% of the sensors are randomly selected once as unknown targets before training and then fixed across all methods and optimization seeds. These target sensors are excluded from both the training inputs and reconstruction loss. The observations are chronologically split into 70% for training, 10% for validation, and 20% for testing. The validation MAE is used for early stopping and checkpoint selection, after which the selected model is evaluated once on the test split. All experiments were repeated 5 times and average results were reported.

Evaluation metrics. We report mean absolute error (MAE), root mean squared error (RMSE), and mean relative error (MRE), where MRE is the sum of absolute errors divided by the sum of absolute ground-truth values over valid entries.

Implementation details. All of our experiments were conducted on NVIDIA GeForce RTX 5070Ti graphics cards. The input window and batch size are 24 and 8. We train STC-CPR with Adam at a learning rate of 5×10^{-4}. During training, additional random node masks are sampled only from the remaining observed sensors, and the reconstruction loss is evaluated on these artificially hidden sensors. Depending on the dataset, we use two or three layers, $d = 16$–64, $K = 30$–60, and $M_p = 15$–30. Complete configurations and baseline settings are provided with the code.

Comparison Experiments

We compare STC-CPR with five representative baselines: KCN (Appleby, Liu, and Liu 2020), IGNNK (Wu et al. 2021), GRIN (Cini, Marisca, and Alippi 2022), OPCR (Deng et al. 2024), and DarkFarseer (Liang et al. 2026), covering classical kriging convolution, inductive graph learning, graph-recurrent imputation, sparse spatio-temporal learning, and robust spatio-temporal kriging.

Table 1 shows that STC-CPR achieves the strongest overall imputation performance across the four datasets. Under the 30% unknown-node setting, it ranks first for all twelve dataset–metric combinations. Compared with the strongest baseline on each dataset, STC-CPR reduces MAE by 2.1%–5.5%, demonstrating consistent gains on both traffic-flow and traffic-speed imputation. Under the more challenging 50% setting, STC-CPR obtains the best result in eleven of the twelve comparisons: it achieves the lowest MAE and MRE on every dataset and the lowest RMSE on PEMS08, METR-LA, and PEMS-BAY. On PEMS04, its RMSE of 110.38 ranks second to DarkFarseer, while its MAE and MRE remain the best. Overall, these results indicate that compact spatio-temporal representation and prototype-mediated interaction can effectively reconstruct completely unobserved sensors across traffic networks with different scales, signal types, and missing rates.

Visualization

We randomly select one target sensor from the unknown-node set for visualization. Figure 3 compares STC-CPR with IGNNK, OPCR and DarkFarseer over a contiguous test interval. STC-CPR follows the overall level and temporal evolution of the ground truth more closely, particularly during the morning increase and evening decline, indicating that the compact latent field preserves both long-term trends and short-term variations for an entirely unobserved sensor.

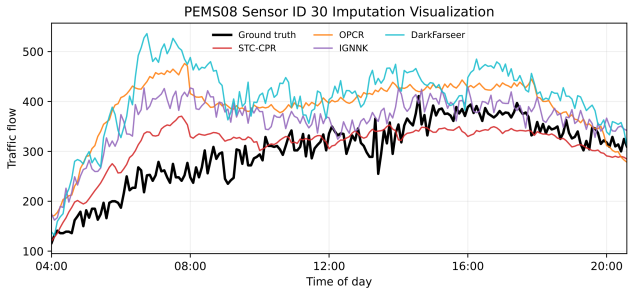


Figure 3: Imputation visualization for a randomly selected PEMS08 target sensor under the 50% unknown-node setting.

Method	0	0.1	0.2	0.3	0.5
KCN	91.17	91.27	91.56	92.05	93.53
IGNNK	81.47	81.65	82.19	83.07	85.75
GRIN	91.22	91.25	91.29	91.35	91.52
OPCR	84.36	84.36	84.35	84.34	84.31
DarkFarseer	80.12	80.60	81.35	81.52	84.98
STC-CPR w/o Cons.	78.04	78.22	78.59	79.21	79.89
STC-CPR	**77.57**	**77.58**	**77.63**	**77.83**	**78.40**

Table 2: MAE under visible-node test noise on PEMS08 with 50% unknown nodes. Results are averaged over five noise seeds.

Robustness Analysis

We evaluate robustness on PEMS08 with 50% unknown nodes. All checkpoints are trained and selected on clean data. At test time, Gaussian noise is added only to permanently visible nodes, with standard deviation α times that of the clean visible-node training values; unknown nodes and their targets remain clean. "STC-CPR w/o Cons." denotes the variant trained without the prototype-consistency loss $\mathcal{L}_{\text{feat}}$, while keeping all other components and settings unchanged. Table 2 reports MAE averaged over five fixed noise seeds shared by all methods.

STC-CPR achieves the lowest MAE at every noise level. Removing consistency regularization causes only a modest increase on clean inputs but its effect becomes clearer under stronger noise. At $\alpha = 0.5$, the ablated variant degrades by 2.37%, compared with 1.07% for the full model. Thus, the consistency objective mainly improves robustness to input perturbations. DarkFarseer, IGNNK, and KCN degrade by 6.06%, 5.25%, and 2.59%, respectively. GRIN changes little because its recurrent graph aggregation may smooth input perturbations, although its consistently high error suggests possible over-smoothing. OPCR improves by 0.06%, which is likely due to noise variation or a slight compensation of prediction bias rather than a meaningful benefit from corrupted inputs. Overall, STC-CPR combines the best accuracy with strong robustness to noise.

Efficiency Analysis

We compare the average training time per epoch and peak allocated GPU memory under the 50% unknown-node setting.

Method	PEMS04		PEMS08	
	Time	Memory	Time	Memory
IGNNK	2.10	188	1.40	99
GRIN	31.66	4484	29.76	2514
OPCR	7.58	1328	5.32	804
DarkFarseer	17.94	454	12.11	318
STC-CPR	3.79	773	2.17	366

Table 3: Efficiency comparison on PEMS04 and PEMS08. Time denotes the average training time per epoch in seconds, and memory denotes peak allocated GPU memory in MB.

Table 3 shows that STC-CPR achieves relatively low training time on both datasets and uses less than 1 GB of GPU memory. IGNNK remains faster and more memory-efficient because it directly treats each temporal window as a node feature vector and employs only three diffusion graph-convolution layers. In contrast, STC-CPR embeds each node-time observation into d hidden channels and maintains a latent field for compression and prototype routing. This additional representation cost explains the gap to IGNNK, while STC-CPR is still faster than all other baselines and provides the best imputation accuracy, demonstrating a favorable accuracy–efficiency trade-off.

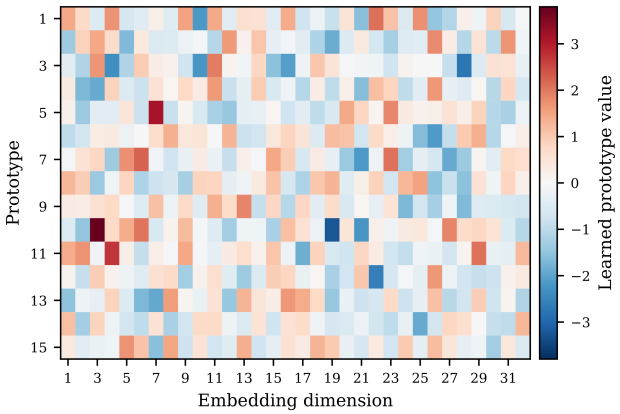


Figure 4: Learned prototype embeddings on PEMS08. Rows and columns denote prototypes and embedding dimensions, respectively.

Ablation Study

We conduct ablations on PEMS04 and METR-LA under the 50% unknown-node setting to examine the contributions of compact-field construction and prototype-mediated interaction. *w/o ST Compression* bypasses both the learnable sensor-to-component decomposition and temporal coarsening, so the remaining global branch operates at the original node–time resolution. *w/o Prototype Routing* retains spatial and temporal compression but replaces the prototype write–read module with an identity mapping. *Prototype-to-AdaGCN* instead replaces prototype routing with an adaptive graph convolution among the K compressed spatial components. This control constructs an adjacency matrix from learnable com-

Variant	PEMS04			METR-LA		
	MAE	RMSE	MRE	MAE	RMSE	MRE
w/o ST Comp.	78.23	111.04	36.75%	6.47	9.79	11.02%
w/o Proto. Routing	79.64	115.23	37.41%	6.59	10.14	11.35%
Proto.-to-AdaGCN	79.02	114.63	37.12%	6.37	9.98	10.85%
STC-CPR	76.96	110.38	36.15%	6.28	9.68	10.70%

Table 4: Ablation study on PEMS04 and METR-LA under the 50% unknown-node setting.

ponent embeddings and applies propagation followed by one pointwise projection. All other configurations are kept identical to the full model.

Table 4 shows that the full STC-CPR consistently achieves the lowest error across both datasets and all three metrics. Removing spatio-temporal compression increases MAE by 1.27 on PEMS04 and 0.19 on METR-LA, demonstrating that constructing a compact field benefits reconstruction on traffic networks with different scales and signal types. Removing prototype routing causes the largest overall degradation: compared with the full model, MAE increases by 2.68 on PEMS04 and 0.31 on METR-LA, accompanied by clear increases in RMSE and MRE. Replacing the prototype router with AdaGCN recovers part of this loss relative to using no interaction module, but it remains inferior to the full model for every metric. These results indicate that interaction among compressed components is useful, while prototype-mediated write–read routing captures reusable traffic patterns more effectively than a simple one-hop adaptive graph in the compact space.

Prototype Analysis

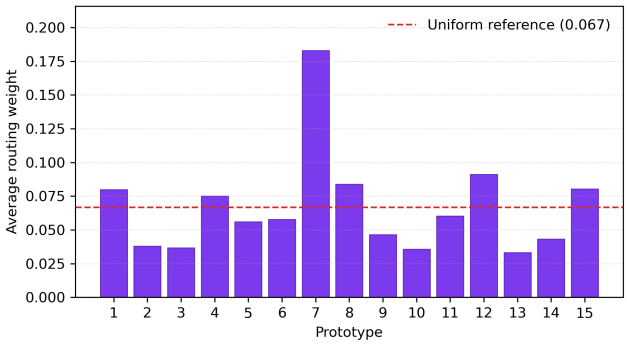


Figure 5: Prototype utilization on PEMS08 with 50% unknown nodes. Statistics are averaged over all test batches, compressed spatial components, time steps, and stacked routing layers. The dashed line denotes uniform utilization as a reference.

Figure 4 exhibits distinct positive and negative patterns across prototypes, with both concentrated and distributed responses. These heterogeneous signatures indicate complementary latent bases rather than a collapsed shared representation. Figure 5 further shows that all 15 prototypes remain active, with average routing weights ranging from 3.30% to 18.29%. The normalized utilization entropy, $-\sum_m u_m \log u_m / \log M_p$, is 0.955, corresponding to an effective prototype count of 13.28 out of 15. Thus, routing makes broad use of the prototype bank while retaining non-uniform specialization.

Hyperparam.	Value	MAE	RMSE	MRE
Layers L	**2**	77.57	103.09	33.25%
	3	77.86	103.12	33.37%
	4	77.67	103.19	33.29%
	5	77.77	103.41	33.34%
Embedding d	16	78.43	104.71	33.85%
	32	77.57	103.09	33.25%
	48	78.35	103.80	33.58%
	64	78.29	103.92	33.56%
Components K	15	77.75	103.92	33.33%
	30	77.57	103.09	33.25%
	45	77.70	103.17	33.31%
	60	77.99	103.77	33.40%
Prototypes M_p	5	77.61	103.19	33.28%
	10	77.77	103.52	33.29%
	15	77.57	103.09	33.25%
	20	77.67	103.77	33.28%

Table 5: Sensitivity analysis on PEMS08 with 50% unknown nodes. Bold marks the default setting.

Sensitivity Analysis

We vary the number of layers L, embedding dimension d, latent spatial components K, and prototypes M_p one at a time on PEMS08 under the 50% unknown-node setting.

Table 5 shows that the three evaluation metrics remain within narrow ranges as each hyperparameter varies, with relative fluctuations consistently below approximately 2%. No sharp or monotonic performance deterioration is observed. This stable behavior indicates that STC-CPR is not sensitive to moderate changes in model depth, representation width, the number of compressed components, or the number of prototypes.

Conclusion

We presented STC-CPR for unknown-node traffic imputation, addressing the high cost of full node–time dependency modeling and the limited reliability of direct interaction over noisy road graphs. STC-CPR learns global dependencies in a compact spatio-temporal field, captures reusable traffic patterns through consistency-regularized prototype routing, and combines them with lightweight bidirectional propagation for node-specific reconstruction. Experiments on four real-world datasets demonstrate strong imputation accuracy, robustness, and computational efficiency.

References

Appleby, G.; Liu, L.; and Liu, L.-P. 2020. Kriging Convolutional Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3187–3194.

Bahadori, M. T.; Yu, Q. R.; and Liu, Y. 2014. Fast Multivariate Spatio-Temporal Analysis via Low Rank Tensor Learn-

ing. In *Advances in Neural Information Processing Systems*, volume 27, 3491–3499.

Bai, L.; Yao, L.; Li, C.; Wang, X.; and Wang, C. 2020. Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting. In *Advances in Neural Information Processing Systems*, volume 33, 17804–17815.

Cao, W.; Wang, D.; Li, J.; Zhou, H.; Li, L.; and Li, Y. 2018. BRITS: Bidirectional Recurrent Imputation for Time Series. In *Advances in Neural Information Processing Systems*, volume 31.

Chen, C.; Petty, K. F.; Skabardonis, A.; Varaiya, P.; and Jia, Z. 2001. Freeway Performance Measurement System: Mining Loop Detector Data. *Transportation Research Record*, 1748(1): 96–102.

Chen, X.; and He, K. 2021. Exploring Simple Siamese Representation Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15750–15758.

Chen, X.; Lei, M.; Saunier, N.; and Sun, L. 2022. Low-Rank Autoregressive Tensor Completion for Spatiotemporal Traffic Data Imputation. *IEEE Transactions on Intelligent Transportation Systems*, 23(8): 12301–12310.

Cini, A.; Marisca, I.; and Alippi, C. 2022. Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks. In *Proceedings of the International Conference on Learning Representations*.

Cover, T.; and Hart, P. 1967. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, 13(1): 21–27.

Cressie, N.; and Wikle, C. K. 2011. *Statistics for Spatio-Temporal Data*. John Wiley & Sons.

Deng, L.; Lian, D.; Wu, C.; and Chen, E. 2024. Learning from Highly Sparse Spatio-temporal Data. In *Advances in Neural Information Processing Systems*, volume 37, 94022–94046.

Du, W.; Côté, D.; and Liu, Y. 2023. SAITS: Self-Attention-Based Imputation for Time Series. *Expert Systems with Applications*, 219: 119619.

Hu, H.; Wen, Z.; Qian, S.; and Co, J. 2025. PAST: A Primary-Auxiliary Spatio-Temporal Network for Traffic Time Series Imputation. arXiv:2511.13414.

Jagadish, H. V.; Gehrke, J.; Labrinidis, A.; Papakonstantinou, Y.; Patel, J. M.; Ramakrishnan, R.; and Shahabi, C. 2014. Big Data and Its Technical Challenges. *Communications of the ACM*, 57(7): 86–94.

Krige, D. G. 1951. A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6): 119–139.

Lee, H.; Jin, S.; Chu, H.; Lim, H.; and Ko, S. 2022. Learning to Remember Patterns: Pattern Matching Memory Networks for Traffic Forecasting. In *Proceedings of the International Conference on Learning Representations*.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *Proceedings of the International Conference on Learning Representations*.

Liang, Z.; Li, W.; Zhang, D.; Jia, Z.; Chen, Y.; Wang, Z.; Zheng, X.; and Youssef, M. 2026. DarkFarseer: Robust Spatio-Temporal Kriging Under Graph Sparsity and Noise. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, 23451–23459.

Lu, G. Y.; and Wong, D. W. 2008. An Adaptive Inverse-Distance Weighting Spatial Interpolation Technique. *Computers & Geosciences*, 34(9): 1044–1055.

Wang, Y.; Peng, H.; Wang, S.; Du, H.; Liu, C.; Wu, J.; and Wu, G. 2025. STAMImputer: Spatio-Temporal Attention MoE for Traffic Data Imputation. Accepted by IJCAI 2025, arXiv:2506.08054.

Wu, Y.; Zhuang, D.; Labbe, A.; and Sun, L. 2021. Inductive Graph Neural Networks for Spatiotemporal Kriging. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4478–4485.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; and Zhang, C. 2020. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 753–763.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; and Zhang, C. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 1907–1913.

Yu, B.; Yin, H.; and Zhu, Z. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 3634–3640.