

## A Appendix

### A.1 Detailed Dataset Descriptions

Detailed description of the six benchmark datasets used in the experiments are provided in Table 1. The METRLA and PEMS BAY datasets provide traffic speed data collected from Los Angeles and the Bay Area, respectively. The PEMS D7(M), PEMS04, PEMS07, and PEMS08 datasets consist of traffic speed and flow records sourced from California’s Performance Measurement System (PeMS)<sup>1</sup>. A brief introduction to PeMS: It is Managed by the California Department of Transportation and provides data from 2001 to the present, collected from approximately 40,000 individual detectors deployed across California’s freeway network in major metropolitan areas. These datasets offers a temporal resolution of 5 minutes, aggregated from original 30-second raw measurements and distributed in csv format.

Table 1: Detailed dataset descriptions.

Dataset	#Sensors (N)	#Timesteps	Time Range	Frequency	Information
METRLA	207	34,272	03/2012 - 06/2012	5min	Traffic Speed
PEMS BAY	325	52,116	01/2017 - 06/2017	5min	Traffic Speed
PEMS D7(M)	228	12,672	05/2012 - 06/2012	5min	Traffic Speed
PEMS04	307	16,992	01/2018 - 02/2018	5min	Traffic Flow
PEMS07	883	28,224	05/2017 - 08/2017	5min	Traffic Flow
PEMS08	170	17,856	07/2016 - 08/2016	5min	Traffic Flow

### A.2 Illustration of *Lazy Mode*

The *lazy mode* refers to a collapse phenomenon in SSDL, where all query representations become overly similar and are mapped to the same prototype. This undermines the discriminative power of the model and limits its ability to capture diverse deviation patterns. In ST-SSDL, this issue arises when the gradient from self-supervised losses directly flows through both the query and prototype representations, allowing the model to minimize objectives trivially without learning meaningful structure.

To address this, we apply the stop-gradient operator  $\tilde{\nabla}(\cdot)$  to the query terms in both the contrastive loss  $\mathcal{L}_{\text{Con}}$  and the deviation loss  $\mathcal{L}_{\text{Dev}}$ . In the contrastive loss:

$$\mathcal{L}_{\text{Con}} = \max \left( \|\tilde{\nabla}(Q^t) - \mathcal{P}^t\|_2^2 - \|\tilde{\nabla}(Q^t) - \mathcal{N}^t\|_2^2 + \delta, 0 \right),$$

and in the deviation loss:

$$\mathcal{L}_{\text{Dev}} = \left\| \tilde{\nabla}(\|Q^t - Q^a\|_1) - \|\mathcal{P}^t - \mathcal{P}^a\|_1 \right\|_1,$$

the operator freezes the query during optimization, forcing the prototypes to adapt around the fixed distribution of query representations. This prevents the model from collapsing into trivial solutions (e.g., assigning all queries to a single prototype), which would otherwise minimize both loss terms without learning meaningful deviations.

By isolating the optimization to prototypes only, this strategy ensures that the latent space remains diverse and structured. It enables the prototype space to reflect semantic spatio-temporal patterns and maintain sufficient granularity for deviation quantification. Without this design, the model tends to fall into a shortcut solution, hence entering the *lazy mode* and severely degrading performance.

Besides theoretical analysis, we also empirically verify this phenomenon: removing the stop-gradient operation leads to single prototype to be chosen by all queries.

<sup>1</sup><https://pems.dot.ca.gov/>

### 32 A.3 Pseudo-code of ST-SSDL

---

**Algorithm 1** ST-SSDL:Spatio-Temporal Time Series Forecasting with Self-Supervised Deviation Learning

---

**Require:** Current Input  $X^t \in \mathbb{R}^{T \times N \times C}$ ; corresponding historical anchor  $X^a \in \mathbb{R}^{T' \times N \times C}$ ; input length  $T$ ; output length  $T'$ ; number of variable  $N$ ; number of prototypes  $M$ ; dimension of latent representation  $h$ ; hidden dimension of query and prototypes  $d$ ;

**Ensure:** Forecasted sequence  $\hat{X}^t$ , loss  $\mathcal{L}$

- 1: **// Encode input and anchor**
- 2:  $H^t \leftarrow GCRU^{\text{enc}}(X^t)$   $\triangleright H^t \in \mathbb{R}^{N \times h}$
- 3:  $H^a \leftarrow GCRU^{\text{enc}}(X^a)$   $\triangleright H^a \in \mathbb{R}^{N \times h}$
- 4: **// Project into query space**
- 5:  $Q^t \leftarrow \text{Linear}(H^t)$   $\triangleright Q^t \in \mathbb{R}^{N \times d}$
- 6:  $Q^a \leftarrow \text{Linear}(H^a)$   $\triangleright Q^a \in \mathbb{R}^{N \times d}$
- 7: **// Compute query-prototype attention score**
- 8:  $\alpha^t \leftarrow \text{softmax}(Q^t \cdot \mathbf{P}^\top / \sqrt{d})$   $\triangleright \alpha^t \in \mathbb{R}^{N \times M}$
- 9:  $\alpha^a \leftarrow \text{softmax}(Q^a \cdot \mathbf{P}^\top / \sqrt{d})$   $\triangleright \alpha^a \in \mathbb{R}^{N \times M}$
- 10: **// Retrieve top-2 prototypes**
- 11:  $\mathcal{P}^t, \mathcal{N}^t \leftarrow \text{Top2}(\alpha^t)$
- 12:  $\mathcal{P}^a, \mathcal{N}^a \leftarrow \text{Top2}(\alpha^a)$
- 13: **// Compute contrastive loss**
- 14:  $\mathcal{L}_{\text{Con}} \leftarrow \max(\|\tilde{\nabla}(Q^t) - \mathcal{P}^t\|_2^2 - \|\tilde{\nabla}(Q^t) - \mathcal{N}^t\|_2^2 + \delta, 0)$
- 15: **// Compute deviation loss**
- 16:  $d_q \leftarrow \|Q^t - Q^a\|_1$
- 17:  $d_p \leftarrow \|\mathcal{P}^t - \mathcal{P}^a\|_1$
- 18:  $\mathcal{L}_{\text{Dev}} \leftarrow \|\tilde{\nabla}(d_q) - d_p\|_1$
- 19: **// Decode with adaptive adjacency**
- 20:  $V^t \leftarrow \alpha^t \mathbf{P}, V^a \leftarrow \alpha^a \mathbf{P}$   $\triangleright V^t \in \mathbb{R}^{N \times d}; V^a \in \mathbb{R}^{N \times d}$
- 21:  $H' \leftarrow W[H^t | V^t | H^a | V^a] + b$   $\triangleright H' \in \mathbb{R}^{N \times d}$
- 22:  $\tilde{\mathcal{A}} \leftarrow \text{Softmax}(\text{ReLU}(H' \cdot H'^\top))$   $\triangleright \tilde{\mathcal{A}} \in \mathbb{R}^{N \times N}$
- 23:  $\hat{X}^t \leftarrow GCRU^{\text{dec}}([H^t, V^t], \tilde{\mathcal{A}})$   $\triangleright \hat{X}^t \in \mathbb{R}^{T' \times N \times C}$
- 24: **// Compute final loss**
- 25:  $\mathcal{L}_{\text{MAE}} \leftarrow \frac{1}{NT'} \sum_{i=1}^N \sum_{\tau=1}^{T'} |\hat{X}_{i,\tau}^t - X_{i,\tau}^t|$
- 26:  $\mathcal{L} \leftarrow \mathcal{L}_{\text{MAE}} + \lambda_{\text{Con}} \mathcal{L}_{\text{Con}} + \lambda_{\text{Dev}} \mathcal{L}_{\text{Dev}}$
- 27: **// Return the forecasting result**
- 28: **return**  $\hat{X}^t$

---

### 33 A.4 Limitations

While ST-SSDL demonstrates strong performance across diverse spatio-temporal forecasting tasks, certain aspects remain to be explored. First, the current prototype-based deviation modeling assumes certain number of prototypes, which may not fully capture extremely fine-grained or highly non-stationary patterns. Although this design supports efficient learning and generalization, future work could consider hierarchical prototype structures to further enhance flexibility. Additionally, the framework has primarily been evaluated on spatio-temporal datasets; applying ST-SSDL to domains with fundamentally different dynamics, such as social networks or financial systems, may require task-specific adaptations. Finally, the model’s inference speed is slightly affected by the GCRU architecture, though this overhead remains acceptable in most practical settings.

### 43 A.5 Broader Impact

ST-SSDL has the potential to improve decision-making in domains such as traffic management, urban planning, and environmental monitoring, where accurate predictions can lead to reduced congestion, better resource allocation, and improved public safety. By introducing a self-supervised mechanism for modeling deviations, ST-SSDL further reduces reliance on labeled data and can adapt to changing conditions more robustly. However, like other data-driven forecasting systems, ST-SSDL relies on

historical sensor data, which may reflect biases in urban infrastructure deployment. If such biases are not addressed, models may perform unevenly across regions, potentially reinforcing existing disparities.

## A.6 Notation Summary

For reference, Table 2 summarizes the key notations and their descriptions in this paper.

Table 2: Notation Table	
Symbol	Description
$N$	Number of nodes
$C$	Number of input channels
$h$	Dimension of latent representations $H$
$d$	Dimension of query $Q$ and prototypes $P$
$T, T'$	Length of input and output sequences, respectively
$X^t \in \mathbb{R}^{T \times N \times C}$	Input sequence at current timestep
$X^a \in \mathbb{R}^{T \times N \times C}$	Timestamp-aligned historical anchor sequence
$X^w \in \mathbb{R}^{S \times T^w \times N \times C}$	Segmented training data by week
$\bar{X}^w \in \mathbb{R}^{T^w \times N \times C}$	Weekly historical anchor (averaged)
$H^t, H^a \in \mathbb{R}^{N \times h}$	Encoded latent states of current and historical input
$Q^t, Q^a \in \mathbb{R}^d$	Query vectors derived from $H^t$ and $H^a$
$\{\mathbf{P}_1, \dots, \mathbf{P}_M\} \in \mathbb{R}^{M \times d}$	Learnable prototype vectors in latent space
$\alpha_i$	Attention score between query and $i$ -th prototype
$V$	Attention-weighted representation from prototype pool
$\mathcal{P}^t, \mathcal{N}^t$	Positive and Negative prototypes for $Q^t$
$\mathcal{P}^a, \mathcal{N}^a$	Positive and Negative prototypes for $Q^a$
$\tilde{\mathcal{A}} \in \mathbb{R}^{N \times N}$	Adaptive graph structure computed from latent states
$H'$	Concatenated augmented hidden representation
$r_\tau, u_\tau$	Reset and update gates in GCRU
$c_\tau$	Candidate hidden state in GCRU
$\mathcal{L}_{\text{MAE}}$	Mean Absolute Error loss for prediction accuracy
$\mathcal{L}_{\text{Con}}, \mathcal{L}_{\text{Dev}}$	Contrastive loss and deviation loss
$\lambda_{\text{Con}}, \lambda_{\text{Dev}}$	Loss weighting hyperparameters
$L$	Number of GCRU layers

## A.7 Full Efficiency Test

We report the computational efficiency of ST-SSDL and all baseline models on the six datasets in Table 3, including the number of parameters, training time per epoch, and inference time. All of them are tested on an Intel(R) Xeon(R) Silver 4314 CPU @ 2.40GHz, 320G RAM computing server, equipped with NVIDIA RTX 5000 Ada Generation graphics cards. Moreover, we also provide a detailed descriptions of each baseline method as follows:

- **GRU** [3]: Gated Recurrent Unit, a type of recurrent neural network (RNN) that uses gating mechanisms (an update gate and a reset gate) to manage information flow, making it effective for sequence modeling tasks.
- **STGCN** [11]: spatiotemporal graph convolutional network, a deep learning model for traffic forecasting on spatial-temporal graphs based on graph convolution operation.
- **DCRNN** [6]: Diffusion convolutional recurrent neural network, a neural network for traffic forecasting using diffusion convolution and RNN.
- **GWNet** [9]: GWNet neural network, a CNN-based deep learning model for traffic forecasting using graph convolution layer and wavenet architecture.
- **MTGNN** [10]: Multivariate Time Series Graph Neural Network, a general framework for multivariate time series forecasting that automatically learns uni-directed relations among

variables through a graph learning module, and captures spatial and temporal dependencies using novel mix-hop propagation and dilated inception layers, respectively.

- **AGCRN** [1]: Adaptive graph convolutional recurrent network for traffic forecasting, learning node-specific patterns through node adaptive parameter learning module and data adaptive graph generation module.
- **GTS** [7]: A model for multivariate time series forecasting that learns the underlying graph structure simultaneously with a Graph Neural Network (GNN) when this structure is not explicitly known. It achieves this by framing the problem as learning a probabilistic graph model, optimizing mean performance over a graph distribution that is parameterized by a neural network, enabling differentiable sampling of discrete graphs.
- **STNorm** [5]: Two normalization modules (temporal normalization and spatial normalization) are proposed to refine the high-frequency and local components of the raw data to help the model distinguish between time and space, respectively.
- **STID** [8]: STID tackles the issue of indistinguishable samples in traffic prediction by simply attaching spatial and temporal Identity information to the data. Using basic Multi-Layer Perceptrons (MLPs) with this approach, STID demonstrates that clearly identifying samples in space and time is crucial.
- **ST-WA** [4]: Spatiotemporal aware traffic time series forecasting model, turning spatiotemporal agnostic models into spatiotemporal aware models with encoding time series from different locations into stochastic variables.
- **STDN** [2]: STDN tackles complex traffic prediction by first building a dynamic graph and using spatio-temporal embeddings. Its core innovation is a module that decomposes traffic data into trend-cyclical and seasonal components for each node. An encoder-decoder then uses these components for prediction, achieving superior performance.

Table 3: Efficiency comparison on all 6 datasets.

Model	METRLA ( $N = 207$ )			PEMSBAY ( $N = 325$ )			PEMSD7(M) ( $N = 228$ )		
	#Params	Train	Infer	#Params	Train	Infer	#Params	Train	Infer
GRU	126K	71.96s	3.21s	126K	172.32s	7.62s	126K	27.96s	1.30s
STGCN	246K	81.39s	2.63s	306K	210.90s	6.35s	257K	33.86s	0.97s
DCRNN	372K	529.94s	15.48s	372K	1071.18s	33.03s	372K	854.15s	30.72s
GWNet	309K	101.11s	2.22s	312K	260.98s	5.82s	310K	37.47s	0.88s
MTGNN	405K	62.95s	1.24s	573K	138.64s	2.80s	435K	21.18s	0.47s
AGCRN	752K	87.15s	2.68s	753K	197.93s	5.38s	752K	31.67s	1.02s
GTS	38377K	190.48s	4.86s	58363K	546.96s	14.91s	12158K	54.12s	1.70s
STNorm	224K	64.80s	0.88s	284K	167.36s	2.31s	235K	23.37s	0.34s
STID	118K	12.06s	0.75s	122K	19.25s	0.62s	118K	4.71s	0.21s
ST-WA	375K	113.90s	7.41s	447K	284.30s	19.25s	388K	41.21s	2.78s
STDN	5971K	459.43s	10.39s	6273K	1479.01s	25.91s	6025K	170.05s	4.37s
<b>ST-SSDL</b>	498K	71.84s	8.65s	498K	199.60s	32.40s	181K	49.66s	6.64s

Model	PEMS04 ( $N = 307$ )			PEMS07 ( $N = 883$ )			PEMS08 ( $N = 170$ )		
	#Params	Train	Infer	#Params	Train	Infer	#Params	Train	Infer
GRU	126K	50.87s	2.30s	126K	249.13s	11.05s	126K	31.23s	1.36s
STGCN	297K	63.95s	1.75s	592K	369.55s	7.67s	227K	32.73s	1.22s
DCRNN	372K	269.71s	9.06s	372K	1330.26s	48.46s	372K	211.56s	6.61s
GWNet	311K	72.94s	1.79s	323K	445.99s	11.26s	309K	37.90s	0.92s
MTGNN	548K	38.84s	0.91s	1368K	212.79s	5.13s	353K	28.48s	0.51s
AGCRN	749K	58.13s	1.69s	755K	342.73s	9.89s	150K	38.44s	1.18s
GTS	16305K	100.87s	3.54s	27088K	1270.23s	62.82s	17136K	69.07s	1.79s
STNorm	275K	44.95s	0.74s	570K	279.37s	4.68s	205K	28.06s	0.58s
STID	121K	5.74s	0.26s	140K	12.32s	0.64s	117K	5.42s	0.24s
ST-WA	436K	76.69s	5.70s	786K	678.93s	60.99s	353K	47.88s	3.80s
STDN	6227K	312.12s	7.98s	4480K	693.32s	32.56s	5876K	200.60s	5.71s
<b>ST-SSDL</b>	182K	83.48s	12.08s	379K	870.12s	217.80s	100K	65.81s	7.97s



## 95 A.8 More In-depth Visualizations

96 To further understand the behavior of different models under various real-world conditions, we  
 97 conduct a comprehensive visual comparison of forecasting results across six models: DCRNN,  
 98 STGCN, AGCRN, MTGNN, STDN, and our proposed ST-SSDL on METR-LA dataset. Figure 1, 2,  
 99 3, and 4 presents four representative cases: low deviation, medium deviation, high deviation, and  
 100 partially missing values. Each case highlights specific challenges in spatio-temporal forecasting and  
 demonstrates how different models respond under these conditions.

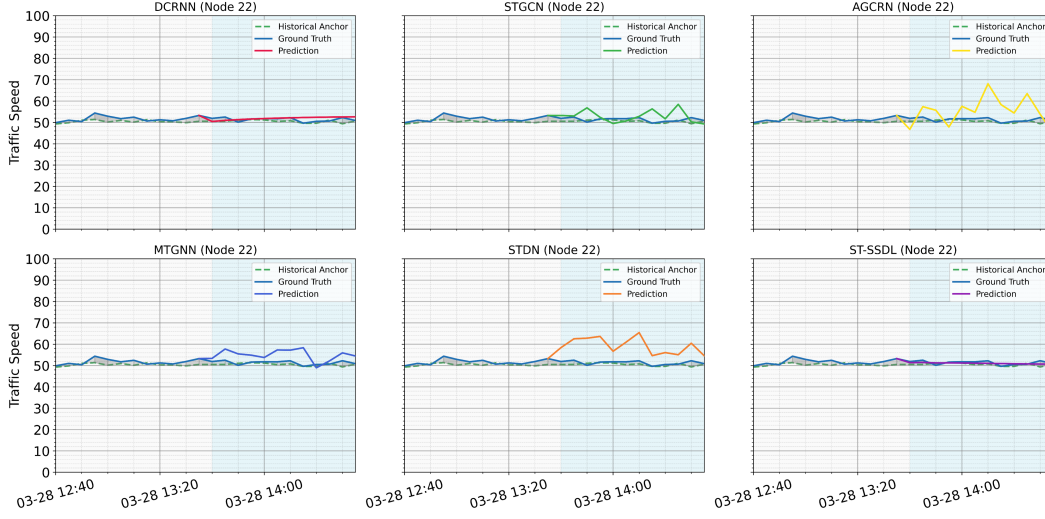


Figure 1: Prediction comparisons under **low deviation**.

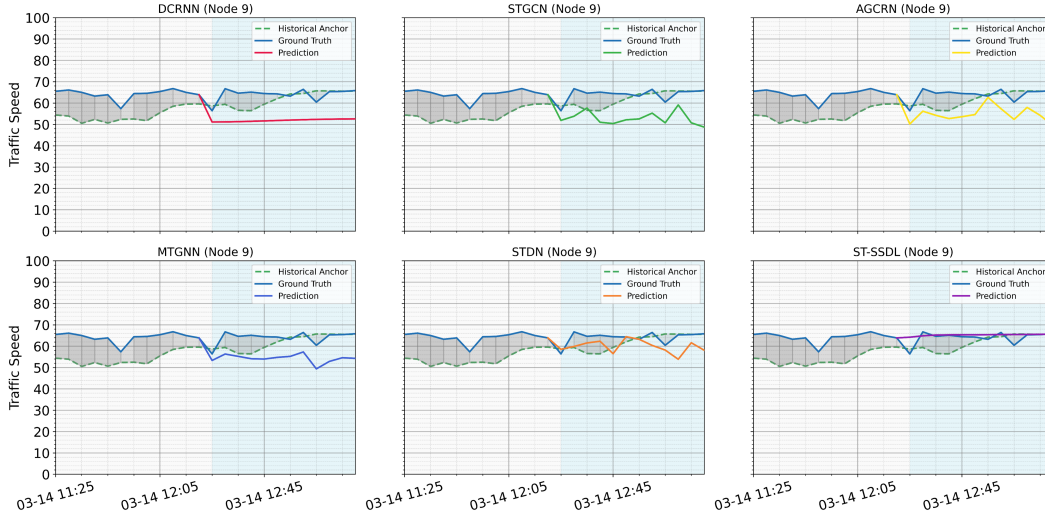


Figure 2: Prediction comparisons under **medium deviation**.

102 **Low-Deviation Scenario.** In this case, the current input remains highly consistent with its historical  
 103 average. Most models are able to make accurate forecasts under this setting. However, we observe  
 104 that AGCRN and STDN exhibit slight discrepancies in the prediction trajectory, possibly due to  
 105 their reliance on static graph structures or insufficient temporal sensitivity. In contrast, ST-SSDL  
 106 maintains precise alignment with the ground truth, demonstrating that it performs competitively even  
 107 in standard settings.

108 **Medium-Deviation Scenario.** When moderate deviation occurs, certain models begin to show  
 109 sensitivity to the changing input dynamics. DCRNN, STGCN, AGCRN, and MTGNN all produce

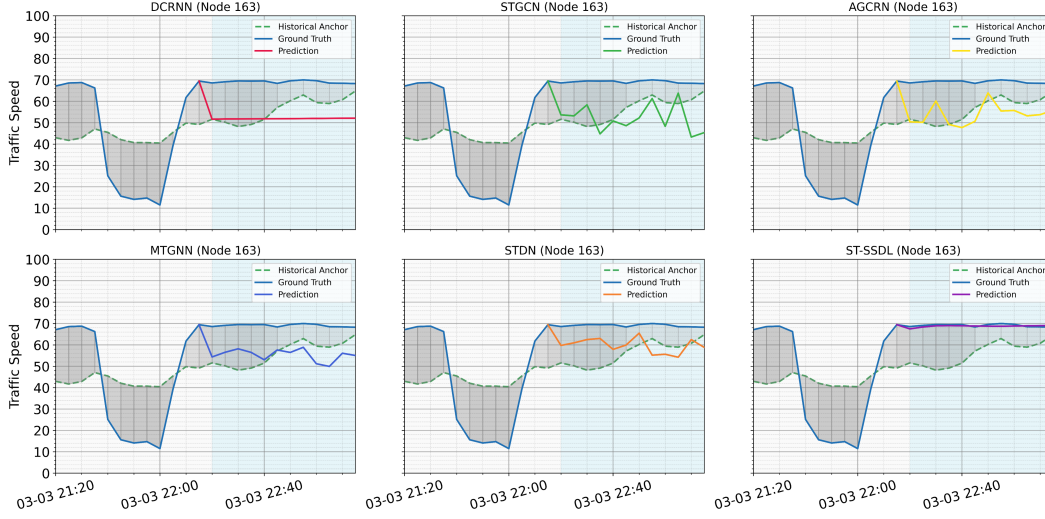


Figure 3: Prediction comparisons under **high deviation**.

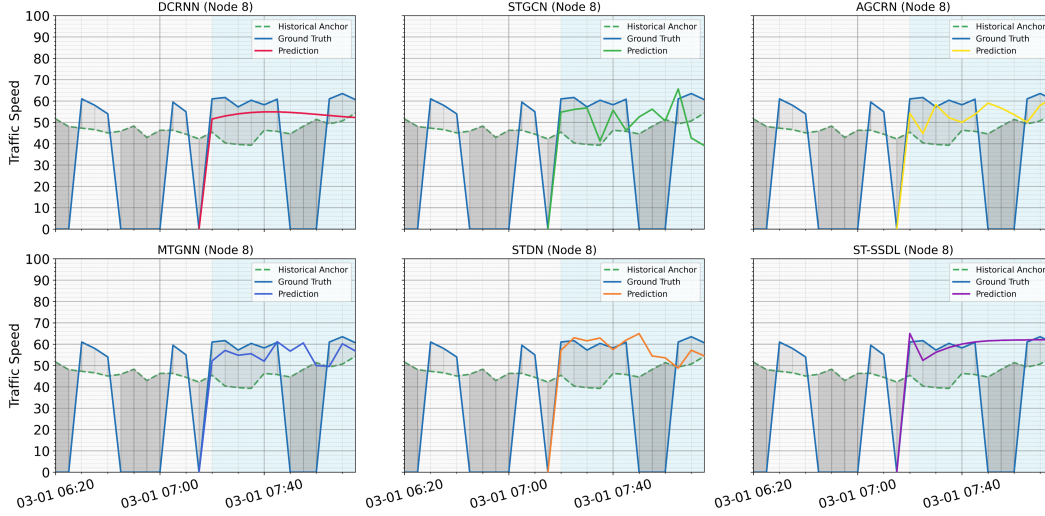


Figure 4: Prediction comparisons under **partially missing values**.

110 predictions that visibly deviate from the ground truth, reflecting a struggle to generalize under  
 111 moderate shifts. ST-SSDL, by contrast, adapts effectively to this condition and provides accurate  
 112 predictions. This supports the effectiveness of our self-supervised deviation modeling strategy, which  
 113 guides the model to dynamically adjust its representation based on contextual shifts.

114 **High-Deviation Scenario.** In this more challenging case, a significant deviation is observed between  
 115 current input and historical trends. All baseline models fail to capture this change and generate  
 116 forecasts that diverge considerably from the actual values. ST-SSDL is the only model that main-  
 117 tains predictive accuracy, closely tracking the ground truth despite the abrupt change. This result  
 118 underscores the necessity of modeling deviation in spatio-temporal forecasting and validates the core  
 119 motivation of our proposed framework.

120 **Partially Missing Values Scenario.** Real-world data often contain missing or incomplete observa-  
 121 tions. In this setting, models such as DCRNN and STGCN suffer performance degradation due to the  
 122 missing input points. ST-SSDL, however, continues to produce reliable predictions. This robustness  
 123 may stem from its ability to interpret missing data as a form of deviation from historical norms,  
 124 allowing it to handle imperfect inputs more effectively. This case highlights an additional advantage  
 125 of deviation modeling: improved resilience to data corruption.

These visual analyses reinforce the motivation and effectiveness of our self-supervised deviation learning framework. ST-SSDL not only excels under large deviations but also maintains robustness in both common and imperfect scenarios.

## References

- [1] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in neural information processing systems*, 33:17804–17815, 2020.
- [2] Lingxiao Cao, Bin Wang, Guiyuan Jiang, Yanwei Yu, and Junyu Dong. Spatiotemporal-aware trend-seasonality decomposition network for traffic flow forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(11):11463–11471, Apr. 2025.
- [3] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [4] Razvan-Gabriel Cirstea, Bin Yang, Chenjuan Guo, Tung Kieu, and Shirui Pan. Towards spatio-temporal aware traffic time series forecasting. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 2900–2913. IEEE, 2022.
- [5] Jinliang Deng, Xiusi Chen, Renhe Jiang, Xuan Song, and Ivor W Tsang. St-norm: Spatial and temporal normalization for multi-variate time series forecasting. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 269–278, 2021.
- [6] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.
- [7] Chao Shang, Jie Chen, and Jinbo Bi. Discrete graph structure learning for forecasting multiple time series. In *International Conference on Learning Representations*, 2021.
- [8] 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*, pages 4454–4458, 2022.
- [9] 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*, pages 1907–1913, 2019.
- [10] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 753–763, 2020.
- [11] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3634–3640, 2018.