A APPENDIX

A.1 ALGORITHM

Algorithm 1: The PromptST Learning Algorithm

Input: The spatial-temporal graph $\mathcal{G}$, the maximum epoch number E, the learning rate η;
Output: Traffic flow or crime rate $\mathbf{H}$ and trained parameters in $\mathbf{\Theta}_f$ of Prompt neural network and $\mathbf{\Theta}_g$ of GNN-based neural network;

Initialize all parameters in $\mathbf{\Theta}_g$ and $\mathbf{\Theta}_f$;
Train the framework PromptST by Equation 3
for $epoch = 1, 2, ..., E$ **do**
  Split the date into train, test and prompt;
  Train the GNN-based pretrain model via train dataset;
  for $\theta_g \in \mathbf{\Theta}_g$ **do**
    $\theta_g = \theta_g - \eta \cdot \frac{\partial \mathcal{L}}{\partial \theta_g}$
  end
end
for $epoch = 1, 2, ..., E$ **do**
  Freeze parameters of GNN-baed pretrain model and update the prompt neural network via Equation 3 via prompt dataset;
  Compute the MAE loss $\mathcal{L}$ following Equation 3;
  Minimize the loss $\mathcal{L}$ by Equation 6 using gradient decent with learning rate η;
  for $\theta_f \in \mathbf{\Theta}_f$ **do**
    $\theta_f = \theta_f - \eta \cdot \frac{\partial \mathcal{L}}{\partial \theta_f}$
  end
end
Return $\mathbf{H}$ and all parameters $\mathbf{\Theta}_g$ and $\mathbf{\Theta}_f$;

The Algorithm 1 section of our framework PromptST presents specific algorithmic specifics. The initialization of all the parameters is the first step, as seen in Algorithm 1. After then, the GNN-based model is trained until it is proficient via updating parameters of $\mathbf{\Theta}_g$. We train the prompt tuning neural network iteratively and fix the GNN-based model. To improve the performance of the prompt tuning neural network, we employ the MAE loss Wu et al. in accordance with earlier studies that are mentioned in the traffic prediction task. With this approach, the MAE loss is determined after E 3. We tune the prompt tuning neural network 6 until it converges. Following all these steps, the procedure ends and returns all $\mathbf{\Theta}_g$ and $\mathbf{\Theta}_f$ parameters.

A.2 EVALUATION METRICS AND EVALUATION PLATFORM

Following existing studies of traffic flow prediction Bai et al. (2020); Li & Zhu (2021); Fang et al. (2021); Chen et al. (2021a;b); Rao et al. (2022); Lan et al. (2022), we adopt three widely utilized metrics namely Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE) as evaluation metrics for traffic prediction of 5 point-based datasets, namely PeMSD04, PeMSD08, PeMSD03, PeMSD07 and PeMSD-Bay in Table 1, and 3 grid-based datasets, namely NYCTaxi, T-Drive and CHIBike. For crime prediction task, we follow the settings in Xia et al. (2022) in terms of Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) metrics on NYC crime and Chicago crime datasets. All methods are implemented in Python 3.9 and PyTorch 1.12.0. The experiments are conducted on a server with 10-cores of Intel(R) Core(TM) i9-9820X CPU @ 3.30GHz 64.0GB RAM and 4 Nvidia GeForce RTX 3090 GPU.

A.3 EFFECTIVENESS

We conducted experiments on grid-based datasets, specifically NYCTaxi, T-Drive, and CHIBike, to evaluate the performance of our model, PromptST, in terms of inflow and outflow predictions. The results are summarized in Table 8. Upon analysis, we observe that our model consistently achieves the best performance across most cases and demonstrates superior performance in the remaining

Table 7: Data Description of 10 Datasets

Traffci Data	Point-based Datasets					Grid-based Datasets		
Datasets	PeMSD04	PeMSD08	PeMSD03	PeMSD07	PeMS-Bay	NYTaxi	CHIBike	TDrive
Sensors	307	170	358	883	325	75 (15 × 5)	270 (15 × 18)	1024 (32 × 32)
Data	16,992	17,856	26,208	28,224	52,116	17,520	4,416	3,600
Interval	5 minutes	5 minutes	5 minutes	5 minutes	5 minutes	30 minutes	30 minutes	60 minutes
Crime Data	NYC-Crimes					Chicago-Crimes		
Time Span	Jan, 2014 to Dec, 2015					Jan, 2016 to Dec, 2017		
Category	Burglary		Robbery			Theft		Battery
Cases	31,779		33,453			124,630		99,389
Categoty	Assult		Larceny			Damage		Assult
Cases	40,429		85,899			59,886		37,972

Table 8: Overall performance of Grid-based Datasets of Traffic Prediction

Datasets	NYCTaxi						T-Drive						CHIBike					
Metrics	inflow			outflow			inflow			outflow			inflow			outflow		
Models	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
STResNet	14.492	14.543	24.050	12.798	14.368	20.633	19.636	17.831	34.890	19.616	18.502	34.597	4.767	31.382	6.703	4.627	30.571	6.559
DMVSTNet	14.377	14.314	23.734	12.566	14.318	20.409	19.599	17.683	34.478	19.531	17.621	34.303	4.687	32.113	6.635	4.594	31.313	6.455
DSAN	14.287	14.208	23.585	12.462	14.272	20.294	19.384	17.465	34.314	19.290	17.379	34.267	4.612	31.621	6.695	4.495	31.256	6.367
DCRNN	14.421	14.353	23.876	12.828	14.344	20.067	22.121	17.750	38.654	21.755	17.382	38.168	4.236	31.264	5.992	4.211	30.822	5.824
STGCN	14.377	14.217	23.860	12.547	14.095	19.962	21.373	17.539	38.052	20.913	16.984	37.619	4.212	31.224	5.954	4.148	30.782	5.779
GWN	14.310	14.198	23.799	12.282	13.685	19.616	19.556	17.187	36.159	19.550	15.933	36.198	4.151	31.153	5.917	4.101	30.690	5.694
STSGCN	15.604	15.203	26.191	13.233	14.698	21.653	23.825	18.547	41.188	24.287	19.041	42.255	4.256	32.991	5.941	4.265	32.612	5.879
STFGNN	15.336	14.869	26.112	13.178	14.584	21.627	22.144	18.094	40.071	22.876	18.987	41.037	4.234	32.222	5.933	4.264	32.321	5.875
STGODE	14.621	14.793	25.444	12.834	14.398	20.205	21.515	17.579	38.215	22.703	18.509	40.282	4.169	31.165	5.921	4.125	30.726	5.698
STGNCDE	14.281	14.171	23.742	12.276	13.681	19.608	19.347	17.134	36.093	19.230	15.873	36.143	4.123	31.151	5.913	4.094	30.595	5.678
STTN	14.359	14.206	23.841	12.373	13.762	19.827	20.583	17.327	37.220	20.443	15.992	37.067	4.160	31.208	5.932	4.118	30.704	5.723
GMAN	14.267	14.114	23.728	12.273	13.672	19.594	19.244	17.110	35.986	18.964	15.788	36.120	4.115	31.150	5.910	4.090	30.662	5.675
TFormer	13.995	13.912	23.487	12.211	13.611	19.522	18.823	16.910	34.470	18.883	15.674	35.219	4.071	31.141	5.878	4.037	30.647	5.638
ASTGNN	**13.844**	**13.692**	**23.177**	12.112	13.602	**19.201**	18.798	16.101	33.870	18.790	15.584	33.998	4.068	31.131	**5.818**	3.981	30.617	5.609
PromptST	14.123	13.762	23.569	**12.103**	**13.316**	19.462	**18.173**	**15.456**	**32.417**	**18.342**	**15.407**	**32.293**	**4.021**	**31.103**	5.875	**3.745**	**29.017**	**5.398**

cases as well. We attribute this success to the following factors: (1) The integration of a prompt tuning neural network, which incorporates Temporal Convolutional Networks (TCN), proves beneficial in capturing temporal features. This ability to capture and leverage temporal information plays a crucial role in accurately predicting traffic flows. (2) Our model utilizes a residual paradigm, where the initial data is added to the model. This approach ensures that our model maintains the same data distribution as the input of the pre-trained model. This helps to preserve the integrity of the data and contributes to the improved performance of our model. By leveraging these strategies, our model PromptST demonstrates superior performance in traffic flow predictions. The incorporation of the prompt tuning neural network and the residual paradigm effectively capture temporal features and maintain data distribution, respectively, resulting in enhanced prediction accuracy.

A.4 Hyperparameter Study

We conducted a hyperparameter study on four datasets: PeMSD04, PeMSD08, Chicago and NYC crime datasets. The study aimed to investigate the impact of two hyperparameters on model performance: the dimension, ranging from 16 to 128, and the kernel size, ranging from 5 to 11. The evaluation metric used was Mean Absolute Percentage Error (MAPE), and the results are illustrated in Figure 5. Upon analysis, we observed that our model achieved the best performance when the dimension was set to 32 and the kernel size was set to 7. It is worth noting that setting a larger dimension may lead to oversmoothing in the GNN-based backbone model, which can subsequently degrade the performance of the prompt neural network. On the other hand, a larger kernel size may introduce more noise from the traffic data, ultimately reducing the overall performance. By carefully selecting the hyperparameters, we are able to optimize the performance of our model. The findings provide valuable insights for achieving better results in traffic flow predictions.

A.5 Hyperparameter Settings

For fair comparison, all compared algorithms have hidden dimensionality modified from the range [8,16,32,64] to achieve their best performance as reported results at 32. The learning rate η is initialized as 0.003 with weight decay 0.3. For GNN-based models, the number of GCN layer is 3. For prompt tuning network, the number of the TCN Layer is 2 and the number of MLP layer is set as 2. The kernel size of the TCN Layer is set as 7 during which our framework PromptST obtains the best performance from the range of [5,7,9,11]. Following existing settings of traffic prediction, we

Table 9: Comparison of Time of Different Methods (One Week) (Minutes)

Datasets	PeMSD04					PeMSD07				
Models	ASTGCN	STGCN	MTGNN	AGCRN	STSGCN	ASTGCN	STGCN	MTGNN	AGCRN	STSGCN
Time for Training Scratch	73.183	20.564	13.913	28.235	37.899	270.531	60.341	30.905	46.031	127.651
Time for Finetune	57.232	17.886	14.620	17.894	34.167	243.172	52.114	40.865	30.303	115.901
Time for Prompt Tuning	50.818	13.675	9.327	12.013	24.733	216.587	37.187	17.220	20.125	70.851
Faster x than Scratch	30.560%	33.500%	32.962%	57.454%	34.740%	19.940%	38.372%	44.281%	56.280%	44.496%
Faster x than Prompt	11.207%	23.543%	36.204%	32.866%	27.611%	10.933%	28.643%	57.861%	33.587%	38.869%

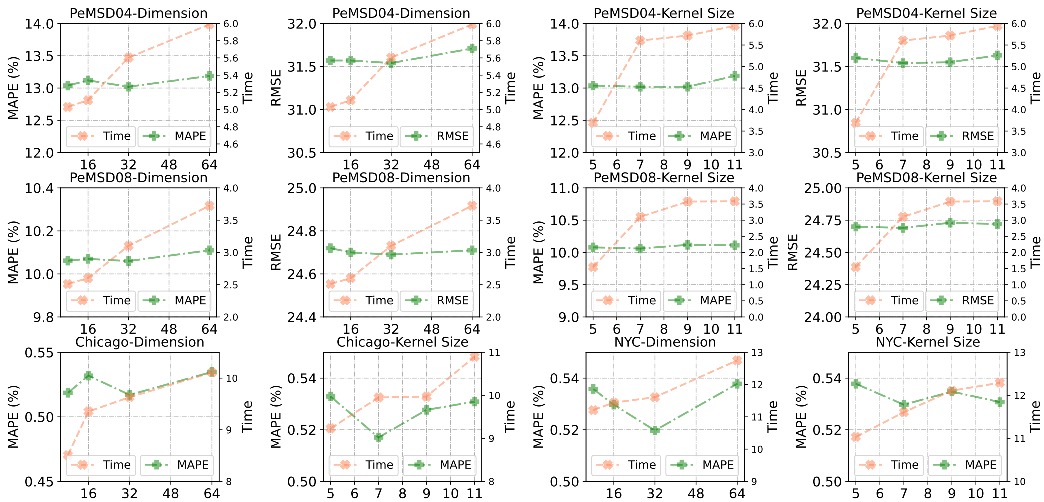

Figure 5: Hyperparameter study on traffic prediction and crime prediction

utilize historical 12 time steps with 5 minutes a step to predict future 12 time steps on point-based datasets (PeMSD04, PeMSD08, PeMSD03, PeMSD07 and PeMS-Bay). And we use historical 6 time steps to predict future 1 time step on grid-based datasets (NYCTaxi, CHIBike and TDrive). All baseline methods follow their predefined settings as their papers.

A.6 Efficiency of Prompt Tuning on Crime Prediction and Traffic Prediction

To evaluate the model's ability to operate independently, we conducted efficiency experiments on traffic predictions using several state-of-the-art baselines. The results are presented in Table 9. From the results, we observed that our prompt tuning neural network significantly improved the efficiency of different baselines, reducing the time cost by approximately 10% to 57%. This finding further validates the advantage of the graph-based passing mechanism in terms of saving time. Additionally, we evaluated the efficiency of crime prediction, as shown in Figure 6. We compared our method's speed in crime prediction to the fine-tuning of a GNN-based model on the New York City and Chicago datasets. The results indicate that our method achieved a speed improvement of 23% to 28% compared to fine-tuning the GNN-based model on the New York City dataset. In the case of the Chicago dataset, our method outperformed fine-tuning by 3% to 10%. These findings highlight the advantage of our PromptST approach in real-life applications, particularly in the field of urban planning. Overall, results demonstrate that our PromptST framework offers improved efficiency across various tasks, making it highly suitable for real-life applications where efficiency is crucial.

A.7 Description of Baselines

We compare 30 baselines including many state-of-art traffic flow prediction methods and crime prediction baselines, where are displayed as following:

- Traffic prediction methods. **DSANet** Huang et al. (2019): It is a method which adopts CNN for capturing temporal correlations and utilizes self-attention mechanism for capturing dynamic spatial information. **DCRNN** Li et al. (2018): To simulate spatial-temporal dependencies, a diffusion convolutional RNN with fusion process is used. **STGCN** Yu et al. (2018): To represent spatial-temporal coupling, it combines a gated temporal convolution module with a graph neural network.

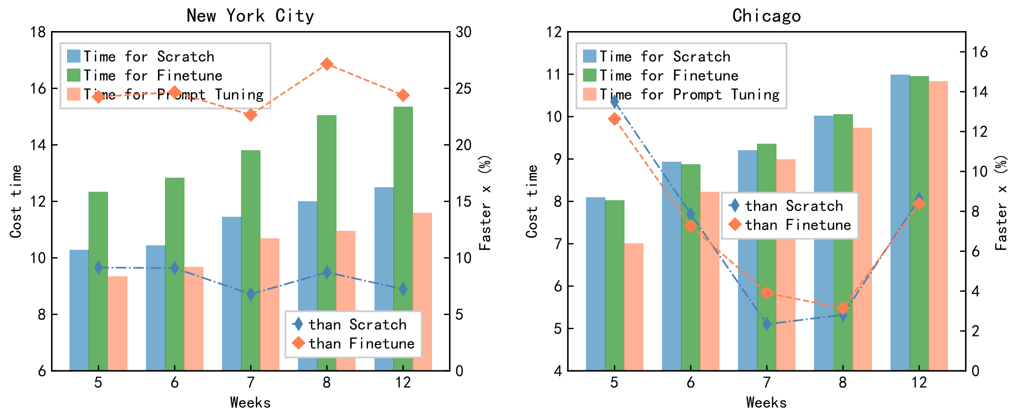

Figure 6: Time-consuming of crime prediction.

GWN Shleifer et al. (2019): It is a technique that combines 1D dilated convolutions and diffusion graph convolutions to capture spatial and temporal changes, enhancing the effectiveness of traffic prediction. **ASTGCN** Zhu et al.: It is an attention-based GCN model that additionally incorporates STGCN for capturing dynamic spatial and temporal information with spatial-temporal attention. **LSGCN** Han & Gong (2022): To capture spatial dynamics, it combines a graph attention network with a graph convolution network. And to capture temporal dynamics, it uses a temporal convolution network. **STSGCN** Song et al. (2020): By stacking numerous localized GCN layers with an adjacent matrix on the time dimension, it captures spatial-temporal correlations. **AGCRN** Bai et al. (2020): In order to capture spatial-temporal correlations, it uses learnt node embeddings in graph convolutions. **STFGNN** Li & Zhu (2021): The performance of traffic prediction is improved by using a spatial-temporal fusion graph neural network to capture spatial-temporal correlations. **STG-ODE** Fang et al. (2021): To address the limitiation caused by the neural networks' lack of depth, it uses differential equations. Shallow GNNs are unable to capture long-range spatial dynamics, and they ignore temporal dynamics, which are crucial for the task of traffic prediction. **Z-GCNETs** Chen et al. (2021a): For predicting traffic flow, it uses zigzag persistence along with a temporal-aware graph convolution network. **TAMP** Chen et al. (2021b): To capture dynamic spatial dependencies, it employs multiple persistence to collect temporal features, which are subsequently fed into graph convolutional networks. **DSTAGNN** Lan et al. (2022): The pre-defined static graph that is typically utilized in classic graph convolution is proposed to be replaced with a new dynamic spatial-temporal aware graph in this study that is based on a data-driven technique. Then, using an improved multi-head attention mechanism, it designs a novel graph neural network architecture that can not only represent dynamic spatial relevance between nodes but also acquire a wide range of dynamic temporal dependency from multiple receptive field features using multi-scale gated convolution. **FOGS** Rao et al. (2022): It is a technique that builds the association graph using the nodes' spatial-temporal dynamics. **STResNet** Zhang et al. (2017): It creates a complete STResNet structure based on the particular characteristics of spatial-temporal data. To describe the temporal closeness, period, and trend characteristics of crowd traffic, it specially uses the residual neural network framework. Based on data, STResNet learns to dynamically aggregate the output of the three residual neural networks. **DMVSTNet** Yao et al. (2018): To model both spatial and temporal relations, it suggests using a Deep Multi-View Spatial-Temporal Network (DMVSTNet) framework. This method specifically consists of three views: temporal, spatial, and semantic. The temporal view models correlations between future demand values with nearby time points using LSTM; the spatial view models local spatial correlation using local CNN. **STGNCDE** Choi et al. (2022): It explains how to use the STGNCDE method, which stands for spatio-temporal graph neural controlled differential equation. The concept of neural controlled differential equations (NCDEs) for processing sequential data is revolutionary. The idea is expanded, and two NCDEs are created: one for spatial processing and the other for temporal processing. **STTN** Xu et al. (2020): To increase the precision of long-term traffic flow forecasting, it suggests a unique paradigm of Spatial-Temporal Transformer Networks (STTNs) that concurrently use dynamical directed spatial dependencies and long-range temporal dependencies. **TFormer** Jin et al. (2023a): It suggests a brand-new model called Trafformer that combines temporal and spatial data into a single transformer-style model. In the spatial-temporal correlation matrix, TFformer enables each node at each timestamp to interact with each other node at each other timestamp in a single step. TFformer can detect intricate spatial-temporal relationships thanks to this design. **ASTGNN** Guo et al. (2021):

It creates a unique self-attention mechanism in the temporal dimension. In addition to enjoying global receptive fields that are advantageous for long-term forecast, it enables the prediction model to catch the temporal dynamics of traffic data. It creates a dynamic graph convolution module for the spatial dimension, using self-attention to capture the spatial correlations.

- Crime prediction methods. **STrans** Wu et al. (2020): By stacking two layers of Transformer to represent spatial-temporal links across spaces and time, it investigates the sparse crimes. For the aggregation of spatial and temporal information, self-attention with query/key transformations is used.. **DeepCrime** Huang et al. (2018): It is a representative baseline for crime prediction that first encodes the temporal embeddings of crime occurrences through time using a recurrent neural network. The next step is to further aggregate temporal representations with the attentional weights using the attention mechanism. **STDN** Yao et al.: A flow gating approach is introduced in this framework to capture the time-aware reliance between areas, and a periodic shifting attention is suggested to learn the temporal patterns between various time periods. **ST-MetaNet** Xu et al. (2018): This model is a meta-learning strategy that uses a GNN-based sequence-to-sequence paradigm to capture various spatial correlations and extract meta information relevant to a given location. **STSHN** Xia et al. (2022): This technique uses hypergraph connections between regions to carry out spatial message transfer between various geographic regions. A stationary approach is taken in building the region hypergraph. Two spatial path aggregation layers are chosen as the number. **DMSTGCN** Han et al. (2021): With the help of this method, the graph convolutional network is enhanced with dynamic and complex geographical and temporal data. The time-aware graph constructor is used to capture relationships between road segments.