A Supplementary Material

In the supplementary material, we provide additional information and details in A.1. This section covers the introduction of data, key parameter settings, comparisons with baselines, optimization methods, and the algorithm process of our method. Furthermore, A.2 presents supplementary experiments for our model, including visualization experiments and replication studies. Additionally, we discuss the reasons behind utilizing hypergraphs as the temporal encoder in A.3. Finally, the limitations and broader impacts of our work are discussed in A.4.

A.1 Data and Implementation Details

Data. The statistical information of the aforementioned four real-world datasets is presented in Table 4. These datasets primarily consist of daily spatio-temporal statistics in the United States. Specifically, the **PeMS08** and **METR-LA** datasets were collected from 7 roads in the San Bernardino area and the road network of Los Angeles County, respectively. On the other hand, the **NYC Taxi** and **NYC Citi Bike** datasets were obtained from New York City.

Parameters. The latent representation dimensionality (d) and the customized parameter (d') are both set to 64 and 16, respectively. The number of hyperedges (H_T, H_S, H_M) is set to 8, 10, and 16, respectively. We perform 2 dynamic routing iterations. Additionally, the balance ratio (λ) for $\mathcal{L}_r$ and $\mathcal{L}_{kl}$ is set to 0.1, and the total mask ratio (r_t) is set to 0.25.

Table 4: Statistical Information of Experimental Datasets.

Dataset	Data Record	# Node	Time Steps	Sample Rate	Sample Date
PEMS08	traffic flow	170	17856	5min	1/Jul/2016 - 31/Aug/2016
METR-LA	traffic speed	207	34272	5min	1/Mar/2012 - 30/Jun/2012
NYC Taxi	taxicab records	266	4368	30min	1/Apr/2016 - 30/Jun/2016
NYC Citi Bike	bike orders	250	4368	30min	1/Apr/2016 - 30/Jun/2016

Baselines. We selected 13 methods as baselines and categorized them into distinct groups:

Hybrid spatio-temporal prediction method:

- **DMVSTNET** [44]: This framework for demand forecasting utilizes RNNs, convolutional networks, and fully connected layers. RNNs capture temporal correlation, convolutional networks handle spatial correlation, and fully connected layers address regional semantic correlation.

Spatio-temporal prediction method based on GNNs:

- **STGCN** [48]: It employs convolution operations to model both temporal and spatial dependencies.
- **GWN** [40]: This method utilizes a learnable graph structure to capture spatial dependencies and incorporates the diffuse convolution technique to capture temporal dependencies.
- **GWN*** [40]: To evaluate the effectiveness of GWN, we utilize long-term time series data as input. Our evaluation focuses on predicting the data for the next hour using the data from the preceding two weeks. We employ a simple linear layer to process the long-term features in this prediction.
- **TGCN** [54]: In this method, both RNNs and GNNs are employed to model temporal dependence and capture spatial correlation, respectively.
- **MTGNN** [39]: This method utilizes a learnable graph structure to model associations between multiple variables. It incorporates dilated convolution and skip connections to jointly capture spatio-temporal dependencies.
- **MSDR** [25]: This model proposes a variant of RNNs to make full use of historical time step information, and combines GNNs to model long-range spatio-temporal dependence.
- **STMGCN** [11]: This method is a spatio-temporal framework designed for demand forecasting. It incorporates region association from various aspects and combines the power of RNNs and GNNs to effectively model both spatial and temporal relations.
- **CCRNN** [46]: This is an approach specifically designed for traffic demand prediction. It leverages multiple layers of GNNs, each assigned with different adjacency matrices, to capture hierarchical spatial associations. RNNs are employed to establish temporal dependencies within the network.

- **STSGCN** [32]: This work captures spatio-temporal correlations by constructing a local spatio-temporal graph, enabling the synchronous modeling of these correlations.
- **STFGNN** [22]: This work proposes a data-driven approach that utilizes a gated convolution method to generate spatio-temporal graphs. By learning spatial and temporal dependencies, the approach effectively captures the correlations within the data.

Attention-based spatio-temporal prediction method:

- **ASTGCN** [13]: It utilizes attention and GNNs to capture spatio-temporal periodic dependencies.
- **STWA** [4]: It integrates location-specific and time-varying parameters into the attention network to effectively capture dynamic spatio-temporal correlations.

Spatio-temporal prediction via differential equation:

- **STGODE** [10]: This method captures spatial dependencies by enhancing GNNs through the use of ordinary differential equations. Additionally, temporal dependencies are modeled using a dilated temporal convolutional network.

A.1.1 Optimization Method

In the GPT-STframework, multiple temporal encoders (Section 4.1) and spatial encoders (Section 4.2) are stacked together to generate the final embeddings. The architecture includes two temporal encoders followed by a spatial encoder, forming a spatio-temporal (ST) encoding block. The final embeddings are generated by passing the data through two spatio-temporal encoding blocks.

During the pre-training phase, the predictions $\hat{\mathbf{Y}} \in \mathbb{R}^{R \times T \times F}$ are computed by applying a linear layer to the hidden dimension. To optimize the parameters, the absolute error loss function is utilized, following the approach employed in prior works such as [55, 1].

$$\mathcal{L}_r = \frac{1}{RTFr_t} \sum_{r=1}^{R} \sum_{t=1}^{T} \sum_{f=1}^{F} |(1 - \mathbf{M}_{r,t,f})(\mathbf{X}_{r,t,f} - \hat{\mathbf{Y}}_{r,t,f})| \tag{9}$$

Let $\mathbf{X}$ represent the input, consisting of F features across R regions in the previous T time slots. The two layers of the MLP network discussed in Sec 4.3 can be formalized as follows:

$$\mathbf{Q}_{r,t} = \sigma(\sigma(\mathbf{E}^p_{r,t}\mathbf{W}^p_r + \mathbf{b}^p_r)\mathbf{W}^p_t + \mathbf{b}^p_t) \tag{10}$$

In the given formulation, $\mathbf{W}^p_r \in \mathbb{R}^{d \times d}$ and $\mathbf{b}^p_r \in \mathbb{R}^d$ represent the region-specific parameters, while $\mathbf{W}^p_t \in \mathbb{R}^{d \times d}$ and $\mathbf{b}t^p \in \mathbb{R}^d$ are the time-dynamic parameters created as in Equation 4. The predictions $q \in \mathbb{R}^{H_s \times R \times T}$ are obtained from $\mathbf{Q}$ through a linear layer followed by a softmax function. The KL divergence loss function $\mathcal{L}_{kl}$ can be formulated as follows:

$$\mathcal{L}_{kl} = \sum_{r=1}^{R} \sum_{t=1}^{T} \sum_{i=1}^{H_S} \bar{c}_{i,r,t} \cdot (\log \bar{c}_{i,r,t} - \log q_{i,r,t}) \tag{11}$$

In this case, $\bar{c} \in \mathbb{R}^{H_s \times R \times T}$ is considered as the ground truth classification result. To balance the contribution of the two loss functions, we introduce a parameter λ to adjust their ratio, as follows:

$$\mathcal{L} = \mathcal{L}_r + \lambda \mathcal{L}_{kl} \tag{12}$$

In the downstream task stage, we utilize the pre-trained embeddings $\boldsymbol{\zeta} \in \mathbb{R}^{R \times T \times d}$ along with the raw data representation $\mathbf{E}' = \bar{\mathbf{X}} \cdot \mathbf{e}_0$ (without any mask operation) as the input for the downstream model. To fuse these two inputs, we employ a simple gated fusion layer proposed by [55], which can be formalized as follows:

$$\mathbf{H} = z \cdot \boldsymbol{\zeta} + (1 - z) \cdot \mathbf{E}; \quad z = \delta(\boldsymbol{\zeta}\mathbf{W}_{h,1} + \mathbf{E}\mathbf{W}_{h,2} + \mathbf{b}_h) \tag{13}$$

In the given formulation, $\mathbf{W}_{h,1}, \mathbf{W}_{h,2} \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_h \in \mathbb{R}^d$ are learnable parameters. The variable z represents the gate operation, and $\delta(\cdot)$ denotes the sigmoid activation function. It is important to note that we prevent the backpropagation of $\boldsymbol{\zeta}$ at this stage. By fusing $\boldsymbol{\zeta}$ and $\mathbf{E}'$ using the gated fusion layer, downstream models can leverage the knowledge gained during the pre-training stage for improved predictions. The specific optimization methods employed may vary depending on the downstream models, such as mean absolute error [1, 25] or Huber loss [22, 32].

A.1.2 Algorithm Process

Algorithm 1 represents the process of the adaptive mask strategy described in Section 4.3. On the other hand, Algorithm 2 illustrates the algorithmic process during the pre-training stage.

Algorithm 1: Adaptive Mask Strategy

Input: ST data $\mathbf{X} \in \mathbb{R}^{R \times T \times F}$, maximum epoch number E, total mask ratio r_t, number of $\mathbf{X}$ elements J
Output: mask matrix $\mathbf{M} \in \mathbb{R}^{R \times T \times F}$

for $e = 1$ *to* E **do**
- Calculate the classification results q from $\mathbf{X}$ according to Sec 4.3
- Calculate adaptive mask ratio r_a by $r_a = (e/E)^{\gamma}$
- Calculate the number of masked elements by $m_t = Jr_t$, and then obtain the adaptive masked number m_a and random masked number m_r by $m_a = Jr_a; \;\; m_r = m_t - m_a$
- Randomly select n categories from the classification list until the total number of elements of these categories is greater than m_a. Mask all the elements of the first $n-1$ categories, and randomly mask the elements of the n-th category with the residual masked number in m_a
- Randomly mask the remaining elements with random masked number m_r

end

Algorithm 2: Learning Process of GPT-ST Framework

Input: Spatio-temporal data $\mathbf{X} \in \mathbb{R}^{R \times T \times F}$, mask matrix $\mathbf{M} \in \mathbb{R}^{R \times T \times F}$, dynamic routing iterations number $\mathcal{R}$, maximum epoch number E, learning rate η
Output: Trained parameters in $\mathbf{\Theta}$

Initialize all parameters in $\mathbf{\Theta}$
for $e = 1$ *to* E **do**
- Calculate approximate classification results q according to Eq 10 and then generate the mask matrix $\mathbf{M}$ according to Alg 1
- Mask the elements in $\mathbf{X}$ with $\mathbf{M}$ and then calculate the initial representation $\mathbf{E}$ of the masked traffic data for each region in each time slot
- Calculate the raw temporal features $\mathbf{d}_t$ according to Eq 4 and initialize the free-form region embedding matrices $\mathbf{c}_r$
- Encode the temporal traffic pattern with $\mathbf{\Gamma}$ by integrating customized parameters $\mathbf{d}_t$ and $\mathbf{c}_r$ into the temporal hypergraph neural network according to Eq 3
- Generate the normalized region embedding $\mathbf{v}$ and calculate the transferred information $\bar{\mathbf{v}}_{i|r,t} \in \mathbb{R}^d$ from each region r to each cluster center (hyperedge) i according to Eq 5
- **for** $r_n = 0$ *to* $\mathcal{R}$ **do**
  - Perform the dynamic routing algorithm to characterize the semantic similarities between the regions (low-level capsules) and the spatial cluster centroids (high-level capsules) according to Eq 6
- **end**
- Generate the final cluster embedding $\bar{\mathbf{s}}$ for cross-class relationships learning
- Generate the personalized high-level hypergraph structure and conduct it on the reshaped embedding $\tilde{\mathbf{s}}$ to generate $\hat{\mathbf{s}}$ to model inter-classes relationships according to Eq 7
- Conduct the customized low-level hypergraph structure to propagate the clustered embeddings $\hat{\mathbf{s}}$ back to the regional embeddings $\mathbf{\Psi}$ according to Eq 8
- Make predictions $\hat{\mathbf{Y}}$ and calculate the absolute error loss $\mathcal{L}_r$ according to Eq 9
- Calculate the KL divergence loss $\mathcal{L}_{kl}$ based on Eq 11
- Calculate the final loss $\mathcal{L}$ according to Eq 12
- **for** $\theta \in \mathbf{\Theta}$ **do**
  - $\theta = \theta - \eta \cdot \partial \mathcal{L} / \partial \theta$
- **end**

end
return all parameters $\mathbf{\Theta}$

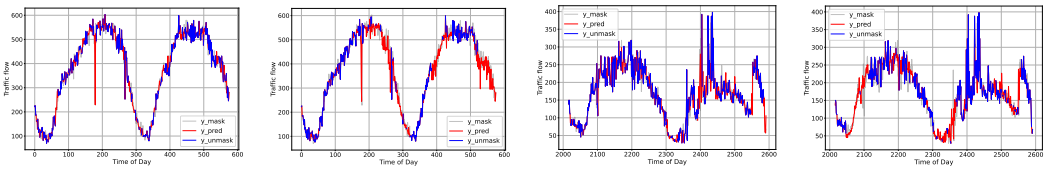
A.2 Additional experiments

A.2.1 Visualization Study

Figure 9 presents the reconstruction results of the model during the pre-training stage. In the figure, gray, red, and blue lines respectively represent the masked signals, predicted values, and visible

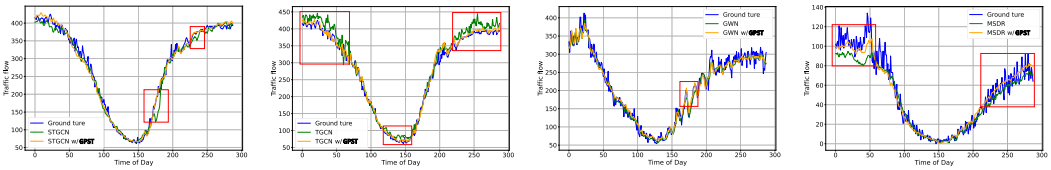
signals. The results demonstrate that GPT-ST can accurately predict the masked signals based on the visible signals, regardless of whether a random mask or an adaptive mask is used. Furthermore, it can be observed that the masked signals generated by the adaptive strategy exhibit greater temporal consistency compared to those generated by the random strategy. This is because the category attribute of a region is less likely to change within a short period of time. The continuous mask and cluster mask are effective in increasing the difficulty of the reconstruction task, enabling GPT-ST to learn robust spatio-temporal representations even with low masking ratios.

Figure 10 summarizes the enhancement effect of your model on downstream baselines. The figure showcases the predicted performance of four baselines on the PEMS08 dataset, where the blue, green, and red lines respectively represent the ground truth, the original performance, and the enhanced performance of the baselines. With the assistance of GPT-ST, the prediction performance of the baseline models experiences significant improvements on certain subsets, as highlighted by the red box. This confirms that the pre-trained model, equipped with customized parameter learners and mechanisms for encoding intra- and inter-class spatial patterns, can provide a discriminative and semantic representation for downstream tasks. As a result, it effectively compensates for the limitations of different baselines.



(a) Case A for Ran mask (b) Case A for Ada mask (c) Case B for Ran mask (d) Case B for Ada mask

Figure 9: Visualization experiments of reconstruction performance in the pre-training stage.



(a) STGCN (b) TGCN (c) GWN (d) MSDR

Figure 10: Visualization experiments of different baselines in the downstream stage.

A.2.2 Replication Study

In this section, the deviation of GPT-ST's performance over random parameter initialization is investigated. The statistical results are presented in Table 5. For each performance evaluation on the PEMS08 dataset, we randomly selected 5 seeds. The results indicate that GPT-ST exhibits strong adaptability to different parameter initialization settings, consistently providing stable and high-quality spatio-temporal representations for downstream models.

Table 5: Replication study on PEMS08 dataset.

Model	MAE	RMSE	MAPE	Model	MAE	RMSE	MAPE
STGCN	17.99±0.14	28.56±0.18	11.22±0.24	GWN	15.37±0.28	24.61±0.26	9.74±0.15
w/ GPT-ST	**16.40±0.16**	**26.27±0.29**	**10.62±0.20**	w/ GPT-ST	**14.77±0.05**	**24.19±0.08**	**9.52±0.07**
ASTGCN	18.05±0.30	27.76±0.47	11.40±0.26	TGCN	21.47±0.04	31.82±0.03	16.06±0.25
w/ GPT-ST	**16.83±0.50**	**26.40±0.71**	**10.78±0.45**	w/ GPT-ST	**17.33±0.08**	**26.50±0.11**	**12.39±0.11**
MTGNN	15.34±0.04	24.55±0.09	9.72±0.04	STSGCN	17.96±0.10	28.19±0.20	11.65±0.17
w/ GPT-ST	**14.96±0.05**	**24.47±0.10**	**9.66±0.04**	w/ GPT-ST	**16.28±0.07**	**26.05±0.09**	**10.51±0.08**
STFGNN	17.20±0.07	27.55±0.18	11.09±0.10	STGODE	17.81±0.06	27.64±0.08	11.33±0.07
w/ GPT-ST	**15.90±0.03**	**25.86±0.06**	**10.42±0.16**	w/ GPT-ST	**15.89±0.16**	**25.05±0.16**	**10.59±0.32**
MSDR	16.52±0.16	25.70±0.19	10.69±0.35	STWA	15.80±0.22	25.16±0.36	10.15±0.20
w/ GPT-ST	**15.96±0.06**	**25.09±0.08**	**10.33±0.08**	w/ GPT-ST	**15.30±0.05**	**24.64±0.12**	**9.99±0.16**

A.3 Analysis of Hypergraph as Temporal Encoder

Existing time encoders in spatio-temporal prediction schemes, such as recurrent neural networks (RNNs), temporal convolutional networks (TCNs), and attention mechanisms, have certain limitations. RNNs are prone to losing long-term information due to the vanishing gradient problem. TCNs can only capture temporal information within a limited neighborhood defined by the size of the convolution kernel. Although attention mechanisms consider the interrelationship between time steps, their computational efficiency decreases significantly when the length of the time series (T) is large, resulting in a time complexity of $O(T^2)$. These limitations hinder the ability of existing approaches to effectively capture and model the temporal dynamics in spatio-temporal data.

To address the aforementioned limitations, we propose to utilize the hypergraph neural network as a temporal encoder for time-dependent modeling. A hypergraph is composed of multiple sets of hyperedges, where each hyperedge serves as a learnable information hub connecting time-step information with different weights. By treating different hyperedges as a means of integrating temporal relationships across different dimensions, the hypergraph neural network offers a flexible control over complexity by adjusting the number of hyperedges. This approach allows for modeling long-term temporal dependencies while maintaining low computational costs. The hypergraph neural network combines the advantages of capturing temporal dynamics effectively and efficiently, making it a promising solution for spatio-temporal prediction tasks.

A.4 Limitations and Broader Impacts

The proposed GPT-ST has demonstrated its effectiveness in improving the prediction performance of downstream models. However, it also has two main limitations: i) Task-specific pre-training: GPT-ST requires pre-training for each specific downstream task. This is because different prediction tasks often have distinct data formats and distributions, necessitating task-specific pre-training. For instance, a GPT-ST pre-trained on task A cannot be directly applied to task B. ii) Increased time cost: Both the pre-training process and the enhancement process in the downstream task stage add to the prediction time cost. These additional computations can impact real-time applications or scenarios with stringent time constraints. In future research, we aim to explore more generalized and versatile spatio-temporal pre-training frameworks, along with lightweight algorithms, to address the task-specific pre-training requirement and further reduce the computational overhead.