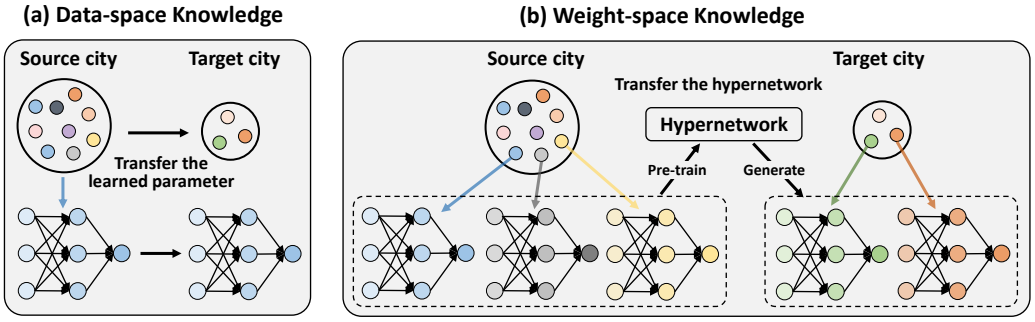

Figure 7: Comparison of data-space and weight-space knowledge across cities. Colorful points represent divided regions in the city.

A Methodology Details

A.1 Different Knowledge for transfer

Figure 7 compares the difference between our framework and other STG transfer learning methods, which focus on weight-space knowledge and data-space knowledge, respectively.

A.2 Region Prompts

We design region prompts to leverage auxiliary data that captures city characteristics. This prompting technique facilitates to utilize external information more flexibly. In our experiments, we adopt a granular approach by crafting distinct prompts for each region within the city. These prompts are meticulously designed to encapsulate the unique characteristics of each region. To facilitate accurate spatio-temporal predictions, we create two specific types of prompts: a spatial prompt and a temporal prompt. The spatial prompt is tailored to provide an in-depth representation of the spatial attributes, encompassing geographical features, environmental conditions, and interconnections with neighboring regions. Complementing the spatial prompt, we also introduce a temporal prompt. This prompt captures the temporal dynamics of each region.

Spatial Prompt. The spatial prompt is obtained by pre-training an urban knowledge graph (UKG) (Zhou et al., 2023), which is meticulously designed to encapsulate the extensive environmental information within a city. Specifically, we represent urban regions as distinct entities within the UKG framework. We use relations "BorderBy" and "NearBy" to capture the spatial adjacency among regions. By leveraging this adjacency representation, we aim to capture the influence that proximate regions exert on one another. Furthermore, our UKG incorporates an understanding of the functional similarity between these urban regions. This insight is quantified by computing the cosine similarity of the distribution of Points of Interest (POI) categories between region pairs. We establish a "SimilarFunc" relation to establish connections between regions that exhibit functional similarity, which emphasizes the critical role played by shared functions in shaping the urban landscape.

Relation	Head & Tail Entity Types	Semantic Information
BorderBy	(Region, Region)	Regions share part of the boundary
NearBy	(Region, Region)	Regions are within a certain distance
SimilarFunc	(Region, Region)	Regions have similar functions

Table 3: The details of relations in the urban knowledge graph.

To extract the spatial prompts from the constructed Urban Knowledge Graph (UKG), we employ a state-of-the-art KG embedding model, TuckER (Balažević et al., 2019), to learn an embedding representation for each region. TuckER evaluates the plausibility of triplets as follows:

$$\phi(h, r, t) = \mathcal{W} \times_1 e_h \times_2 e_r \times_3 e_t, \tag{3}$$

where $\mathcal{W} \in \mathbb{R}^{d^3_{KG}}$ is a learnable tensor, $\times_n$ denotes the tensor product along the n_{th} dimension, and $e_h, e_r, e_t \in \mathbb{R}^d_{KG}$ are the embeddings of head entity h, tail entity t and relation r respectively. The primary objective of the KG embedding model is to maximize the scoring function for triplets that exist in the UKG, thereby preserving the knowledge contained within the UKG.

In summary, our UKG leverages the 'BorderBy' and 'NearBy' relationships to articulate spatial connections and the 'SimilarFunc' relationship to underscore functional parallels between urban regions. The benefits of UKG are twofold. First, it integrates various relationships within the city, allowing the learned embeddings of regions to provide descriptive information about their respective urban environments. Secondly, in contrast to time series data collected by sensors or GPS devices, the features utilized in the Urban Knowledge Graph (UKG) are readily available in all urban areas. This accessibility makes the UKG scalable and adaptable to cities, even those with limited development levels. The details of relations in UKG are shown in Table 3.

Temporal Prompt. The temporal prompt is derived through a strategic application of an unsupervised pre-training model designed for time series, as introduced by Shao et al (Shao et al., 2022). This approach shares similarities with the concept of a Masked AutoEncoder (MAE) (He et al., 2022) for sequence data. Initially, the time series data for each region is subjected to a masking procedure, where random patches within the time series are concealed. Subsequently, an encoder-decoder model is trained on this modified data to reconstruct the original time series. This training process is centered around the objective of reconstructing the complete time series based solely on the partially observable series patches. Given that time series data often exhibits lower information density, a relatively high masking ratio (75%) is employed. This higher masking ratio is crucial for creating a self-supervised learning challenge that encourages the model to capture meaningful temporal patterns from incomplete observations. Upon successful completion of the self-supervised learning phase for time series data, the output of the encoder yields the temporal embeddings. In essence, this method capitalizes on self-supervised learning to extract valuable temporal features from time series data, which can subsequently be used as temporal prompts.

A.3 Conditioning Strategies

Pre-conditioning. "Pre" denotes that the prompt is integrated into the token sequence before being fed into self-attention layers. In this method, we simply add the region prompt p to the token embeddings within the input sequence.

Pre-conditioning with inductive bias. In this variant, we adopt a different approach to add the region prompt p to the token embeddings within the input sequence. The spatial prompt is incorporated into spatial-related parameters uniformly and the temporal prompt into temporal-related parameters uniformly as follows:

$$\begin{aligned}
[\boldsymbol{x}_{s,1}, \boldsymbol{x}_{s,1}, \cdots, \boldsymbol{x}_{s,m}] &= [\boldsymbol{x}_{s,1}, \boldsymbol{x}_{s,1}, \cdots, \boldsymbol{x}_{s,m}] + [\underbrace{\boldsymbol{p}_s, \boldsymbol{p}_s, \cdots, \boldsymbol{p}_s}_{m}] \\
[\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,1}, \cdots, \boldsymbol{x}_{t,n}] &= [\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,1}, \cdots, \boldsymbol{x}_{t,n}] + [\underbrace{\boldsymbol{p}_t, \boldsymbol{p}_t, \cdots, \boldsymbol{p}_t}_{n}],
\end{aligned} \tag{4}$$

where m and n represent the number of tokens for spatial parameters and temporal parameters, p_s denotes the spatial prompt, and p_t denotes the temporal prompt.

Pre-adaptive conditioning. In this variant, we introduce an attention mechanism, which determines to what extent the prompt should be added to specific token embeddings. We denote the prompt as $p \in \mathbb{R}^{2\times E}$, where E is the embedding size of spatial and temporal prompts. This approach aims to empower the model to learn how to adaptively utilize the prompts, enhancing its conditioning capabilities. The utilization of the prompt can be formulated as follows:

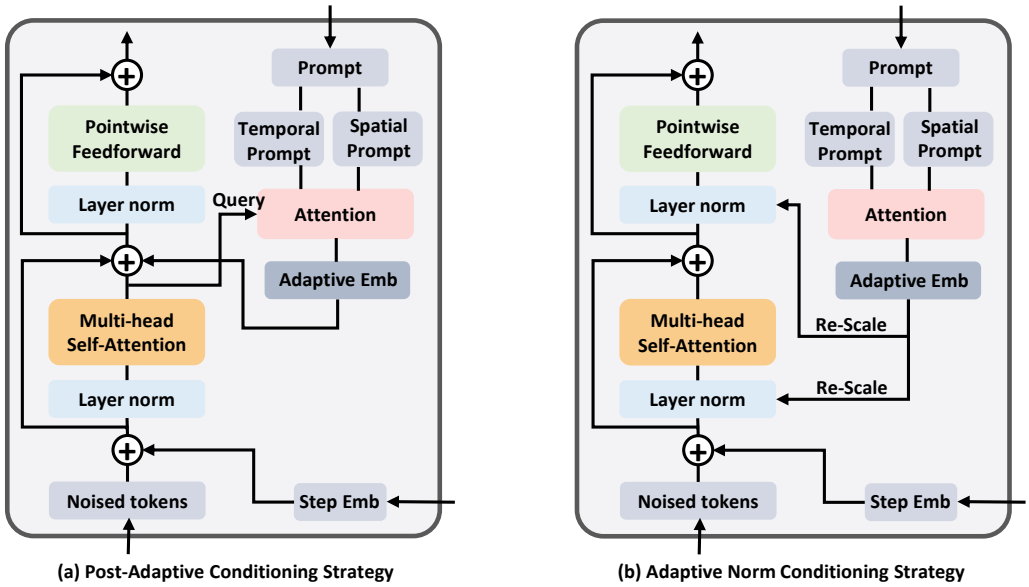

Figure 8: Illustration of two conditioning strategies: (a) "Post-Adaptive Conditioning" and (b) "Adaptive Norm Conditioning".

$$u_j = \tanh(W_w p_j + b_w), j \in \{0, 1\} \tag{5}$$

$$\alpha_{i,j} = \frac{\exp(u_j^T, x_i)}{\sum_k \exp(u_k^T, x_i)} \tag{6}$$

$$P_i = \sum_j \alpha_{i,j} p_j, j \in \{0, 1\} \tag{7}$$

where p_0 and p_1 represent the spatial prompt and temporal prompt, respectively, P_i denotes the aggregated prompt from two aspects for the i_{th} token.

Post-adaptive Conditioning. Figure 8 (a) illustrates this conditioning strategy. The aggregated prompt based on the attention mechanism is added after the multi-head self-attention in each transformer layer. Specifically, the query used for spatio-temporal attentive aggregation is the output of the multi-head self-attention layer.

Adaptive norm conditioning. Figure 8 (b) illustrates this conditioning strategy. The aggregated prompt based on the attention mechanism is used for re-scaling the output in each layer norm.

A.4 Network Layers of Spatio-temporal Prediction Models

We provide details of parameter tokenizers introduced in Section 3.5. In our experiments, we implement our framework on two spatio-temporal prediction models, STGCN (Yu et al., 2017) and GWN (Wu et al., 2019). We present how to transform their network layers into a vector-based token sequence. According to Table 4 and Table 5, the transformation from parameter layers to a token sequence can be formulated as follows:

$$\begin{aligned} gcd &= GCD(numel(s_1), numel(s_2), \ldots, numel(s_m)) \\ L_i &= c_i * numel(s_i)/gcd \\ L &= \sum_i L_i, \end{aligned} \tag{8}$$

where GCD denotes the calculation of the Greatest Common Divisor, m denotes the number of different layer types, s_i denotes the shape, $numel$ denotes the total number of elements in the tensor

	Layer name	Parameter shape	Count
Block1	Block1-TimeBlock1-conv	[32,2,1,4]	3
	Block1-GraphConv_Theta1	[32,8]	1
	Block1-TimeBlock2-conv	[32,8,1,4]	3
TimeBlock	last_TimeBlock-conv	[32,32,1,4]	3
Linear	Fully Connected Layer	[6,96]	1

Table 4: Parameter structure of a STGCN Model

Layer name	Parameter shape	Count
filter_convs.x	$[32, 32, 1, 2]$	8
gate_convs.x	$[32, 32, 1, 2]$	8
residual_convs.x	$[32, 32, 1, 1]$	8
skip_convs.x	$[32, 32, 1, 1]$	8
bn.x	$[32] + [32]$	8
gconv.x.mlp.mlp	$[32, 160, 1, 1]$	8
start_conv	$[32, 2, 1, 1]$	1
end_conv1	$[32, 32, 1, 1]$	1
end_conv2	$[6, 32, 1, 1]$	1

Table 5: Parameter structure of a GWN Model

c_i denotes the count of this type of layer. In this way, we obtain a token sequence with length as L and embedding size as gcd.

STGCN (Yu et al., 2017). Spatio-temporal graph convolution network. Different from regular convolutional and recurrent units, this model build convolutional structures on graphs. We use a 3-layer STGCN block, and utilize a 1-layer MLP as the output predictor. Table 4 shows the detailed network layers of a STGCN.

GWN (Wu et al., 2019). Graph WaveNet. This model developed a novel adaptive dependency matrix and learned it through node embeddings to capture the spatial dependency. It also combines with a stacked dilated causal convolution component. We use a 2-layer 4-block GWN model. Table 4 shows the detailed network layers of a GWN.

A.5 ALGORITHMS

We present the training algorithm for spatio-temporal graph prediction in Algorithm 1. Besides, we present the training algorithm and sampling algorithm for the diffusion model in Algorithm 2 and Algorithm 3, respectively.

Algorithm 1 Model Parameter Preparation

Input: Dataset $D = \{D_1, D_2, \ldots, D_{M_s}\}$, neural networks $F = \{f_{\theta_1}, f_{\theta_2}, \ldots, f_{\theta_{M_s}}\}$ of the spatio-temporal prediction model, loss function L, parameter storage S.
Output: Parameter storage S.
Initialize: Learnable parameters θ_m for f_m, parameter storage $S = \{\}$.
for $m \in \{1, 2, \ldots, M_s\}$ **do**
    for $epoch \in \{1, 2, \ldots, N_{iter}\}$ **do**
        Sample a mini-batch of inputs and labels from the dataset D_m $\{x, y\} \sim D_m$
        Compute the predictions $\hat{y} \leftarrow f_{\theta_m}(x)$
        Compute the loss $\mathcal{L} \leftarrow L(\hat{y}, y)$
        Update the model's parameters $\theta_m \leftarrow update(\mathcal{L}; \theta_m)$
    end for
    Save the optimized model $S \leftarrow S \cup \{\theta_m\}$
end for

Algorithm 2 Hypernetwork Pre-training

Input: Parameter storage S, hypernetwork G.
Initialize: Learnable parameters γ for G
for $epoch \in \{1, 2, \ldots, N_{iter}\}$ **do**
    Sample a parameter sample $\theta_0 \sim q(\theta_0)$ and $\epsilon \sim \mathcal{N}(0, I)$
    Sample diffusion step $k \sim \text{Uniform}(1, \ldots, K)$
    Take gradient descent step on

$$\nabla_\gamma \|\epsilon - \epsilon_\gamma(\sqrt{\overline{\alpha}_k}\theta_0 + \sqrt{1-\overline{\alpha}_k}\epsilon, p, k)\|^2$$

end for

Algorithm 3 Parameter Sampling

Input: Gaussian noise $\theta_K \sim \mathcal{N}(0, I)$, prompts $P = \{p_1, p_2, \ldots, M\}$
Output: Parameters $\theta_t = \{\theta_{1,0}, \theta_{2,0}, \ldots, \theta_{M_t,0}\}$.
Initialize: Learnable parameters γ for G, $\theta_s = \{\}$.
for $m \in \{1, 2, \ldots, M_t\}$ **do**
    for $k = K \to 1$ **do**
        if $K > 1$ **then**
            $z \sim \mathcal{N}(0, I)$
        else
            $z = 0$
        end if

$$\theta_{m,k-1} = \frac{1}{\sqrt{\alpha_k}}(\theta_{m,k} - \frac{\beta_k}{\sqrt{1-\overline{\alpha}_k}}\epsilon_\gamma(\theta_{m,k}, p, k)) + \sqrt{\beta_k}z$$

    end for
    $\theta_s = \theta_s \cup \theta_{m,0}$
end for

B Experiment Details

B.1 Datasets

In this section, we introduce the details of the used real-world datasets.

B.1.1 Crowd Flow Prediction.

- **NYC Dataset.** In this dataset, we define regions as census tracts and aggregate taxi trips from NYC Open Data to derive hourly inflow and outflow information. The division of train and test regions is based on community districts. Specifically, the Manhattan borough comprises 12 community districts, and we designate 9 of them as train regions, reserving the remaining 3 for testing. Region-specific features encompass the count of Points of Interest (POIs), area, and population.
- **Washington, D.C. Dataset.** Regions in this dataset are defined as census tracts, and inflow data is calculated based on Point of Interest (POI)-level hourly visits. The partitioning of train and test regions is done by counties. Specifically, regions within the District of Columbia are selected as train regions, and regions in Arlington County are designated as test regions. This dataset comprises rich region features, including demographics and socioeconomic indicators.
- **Baltimore Dataset.** Similar to the D.C. dataset, regions, and inflow data in the Baltimore dataset are obtained in the same manner. Train regions consist of regions in Baltimore City, while test regions encompass Baltimore County. This dataset includes the same set of features as the D.C. dataset.

B.1.2 Traffic Speed Prediction.

We conducted our performance evaluation using four real-world traffic speed datasets, following the data preprocessing procedures established in prior literature (Li et al., 2018; Lu et al., 2022).

Datasets	New York City	Washington, D.C.	Baltimore
#Nodes	195	194	267
#Edges	555	504	644
Interval	1 hour	1 hour	1 hour
Time span	2016.01.01-2016.06.30	2019.01.01-2019.05.31	2019.01.01-2019.05.31
Mean	70.066	30.871	18.763
Std	71.852	58.953	28.727

Table 6: The basic information and statistics of four real-world datasets for crowd flow prediction.

Datasets	METR-LA	PEMS-BAY	Didi-Chengdu	Didi-Shenzhen
#Nodes	207	325	524	627
#Edges	1722	2694	1120	4845
Interval	5 min	5 min	10 min	10 min
Time span	2012.05.01-2012.06.30	2017.01.1-2017.06.30	2018.1.1-2018.4.30	2018.1.1-2018.4.30
Mean	58.274	61.776	29.023	31.001
Std	13.128	9.285	9.662	10.969

Table 7: The basic information and statistics of four real-world datasets for traffic speed prediction.

To construct the spatio-temporal graph, we treated each traffic sensor or road segment as an individual vertex within the graph. We then computed pairwise road network distances between these sensors. Finally, we construct the adjacency matrix of the nodes using road network distances and a thresholded Gaussian kernel.

- **METR-LA (Li et al., 2018; Lu et al., 2022).** The traffic data in our study were obtained from observation sensors situated along the highways of Los Angeles County. We utilized a total of 207 sensors, and the dataset covered a span of four months, ranging from March 1, 2012, to June 30, 2012. To facilitate our analysis, the sensor readings were aggregated into 5-minute intervals.
- **PEMS-BAY (Li et al., 2018; Lu et al., 2022).** The PEMS-BAY dataset comprises traffic data collected over a period of six months, from January 1st, 2017, to June 30th, 2017, within the Bay Area. The dataset is composed of records from 325 traffic sensors strategically positioned throughout the region.
- **Didi-Chengdu (Lu et al., 2022).** We utilized the Traffic Index dataset for Chengdu, China, which was generously provided by the Didi Chuxing GAIA Initiative. Our dataset selection encompassed the period from January to April 2018 and covered 524 roads situated within the central urban area of Chengdu. The data was collected at 10-minute intervals to facilitate our analysis.
- **Didi-Shenzhen (Lu et al., 2022).** We utilized the Traffic Index dataset for Shenzhen, China, which was generously provided by the Didi Chuxing GAIA Initiative. Our dataset selection included data from January to April 2018 and encompassed 627 roads located in the downtown area of Shenzhen. The data collection was conducted at 10-minute intervals to facilitate our analysis.

B.2 BASELINES

- **HA.** Historical average approach models time series as a seasonal process and leverages the average of previous seasons for predictions. In this method, we utilize a limited set of target city data to compute the daily average value for each node. This historical average then serves as the baseline for predicting future values.
- **ARIMA.** Auto-regressive Integrated Moving Average model is a widely recognized method for comprehending and forecasting future values within a time series.
- **RegionTrans (Wang et al., 2019).** RegionTrans assesses the similarity between source and target nodes, employing it as a means to regulate the fine-tuning of the target. We use STGCN and GWN as its base model.

- **DASTNet (Tang et al., 2022).** Domain Adversarial Spatial-Temporal Network, which undergoes pre-training on data from multiple source networks and then proceeds to fine-tune using the data specific to the target network's traffic.
- **AdaRNN (Du et al., 2021).** This cutting-edge transfer learning framework is designed for non-stationary time series data. The primary objective of this model is to mitigate the distribution disparity within time series data, enabling the training of an adaptive model based on recurrent neural networks (RNNs).
- **MAML (Finn et al., 2017).** Model-Agnostic Meta Learning is an advanced meta-learning technique designed to train a model's parameters in a way that a minimal number of gradient updates result in rapid learning on a novel task. MAML achieves this by acquiring an improved initialization model through the utilization of multiple tasks to guide the learning process of the target task.
- **TPB (Liu et al., 2023).** Traffic Pattern Bank-based approach. TPB employs a pre-trained traffic patch encoder to transform raw traffic data from cities with rich data into a high-dimensional space. In this space, a traffic pattern bank is established through clustering. Subsequently, the traffic data originating from cities with limited data availability can access and interact with the traffic pattern bank to establish explicit relationships between them.
- **ST-GFSL (Lu et al., 2022).** ST-GFSL generates node-specific parameters based on node-level meta-knowledge drawn from the graph-based traffic data. This approach ensures that parameters are tailored to individual nodes, promoting parameter similarity among nodes that exhibit similarity in the traffic data.

B.3 Implementation Details

In the experiments, we set the number of diffusion steps N=500. The learning rate is set to 8e-5 and the number of training epochs ranges from 3000 to 12000. The dimensions of KG embedding and time embedding are both 128. Regarding the spatio-temporal prediction, we use 12 historical time steps to predict 6 future time steps. Our framework can be effectively trained within 3 hours and all experiments were completed on one NVIDIA GeForce RTX 4090.

C Additional Results

	MAE			RMSE		
Model	**Step 1**	**Step 3**	**Step 6**	**Step 1**	**Step 3**	**Step 6**
HA	21.520	21.520	21.520	47.122	47.122	47.122
ARIMA	19.703	15.063	27.083	37.809	35.093	61.675
RegionTrans	12.116	13.538	15.501	27.622	30.999	36.095
DASTNet	11.501	14.255	17.676	25.551	31.466	38.004
MAML	10.831	13.634	16.192	24.455	30.740	36.271
TPB	**9.153**	<u>11.870</u>	15.236	23.420	28.756	33.424
STGFSL	9.636	12.178	<u>14.116</u>	**22.362**	<u>28.287</u>	<u>33.547</u>
Ours	<u>10.339</u>	**11.454**	**13.144**	<u>23.348</u>	**26.798**	**30.959**

Table 8: Performance comparison of few-shot scenarios on Washington D.C. dataset at different prediction steps in terms of MAE and RMSE. Bold denotes the best results and <u>underline</u> denotes the second-best results.

	MAE			RMSE		
Model	**Step 1**	**Step 3**	**Step 6**	**Step 1**	**Step 3**	**Step 6**
HA	15.082	15.082	15.082	26.768	26.768	26.768
ARIMA	11.150	12.344	18.557	19.627	21.665	35.520
RegionTrans	6.782	7.371	7.895	12.454	13.778	14.648
DASTNet	6.454	7.461	8.424	12.304	13.960	15.225
MAML	6.765	8.170	8.834	13.227	14.470	16.953
TPB	6.014	7.322	8.571	9.832	14.512	16.308
STGFSL	5.925	7.244	8.356	11.157	13.450	15.444
Ours	**5.570**	**5.971**	**6.582**	**10.165**	**11.344**	**13.003**

Table 9: Performance comparison of few-shot scenarios on Baltimore dataset at different prediction steps in terms of MAE and RMSE. Bold denotes the best results and underline denotes the second-best results.

	MAE			RMSE		
Model	**Step 1**	**Step 3**	**Step 6**	**Step 1**	**Step 3**	**Step 6**
HA	34.705	34.705	34.705	52.461	52.461	52.461
ARIMA	27.865	27.695	33.771	40.643	45.003	59.359
RegionTrans	16.138	19.318	22.103	26.248	32.489	37.654
DASTNet	16.480	22.464	27.147	25.788	36.717	43.834
MAML	14.083	19.753	24.493	23.473	33.079	40.407
TPB	16.616	21.835	28.005	**20.50**	28.386	37.701
STGFSL	**13.479**	18.654	23.054	21.918	31.106	37.818
Ours	13.580	**16.076**	**18.923**	22.081	**27.329**	**32.260**

Table 10: Performance comparison of few-shot scenarios on NYC dataset at different prediction steps in terms of MAE and RMSE. Bold denotes the best results and underline denotes the second-best results.

	MAE			RMSE		
Model	**Step 1**	**Step 3**	**Step 6**	**Step 1**	**Step 3**	**Step 6**
HA	3.257	3.257	3.257	6.547	6.547	6.547
ARIMA	7.176	7.262	7.114	10.84	11.01	10.89
RegionTrans	2.846	3.278	3.925	4.417	5.729	7.039
DASTNet	2.695	3.205	3.809	4.267	5.516	7.028
MAML	2.756	3.121	3.896	4.182	5.584	6.909
TPB	**2.485**	3.129	3.680	**4.094**	5.513	6.816
STGFSL	2.679	3.187	3.686	4.147	5.599	6.987
Ours	2.587	**3.098**	**3.674**	4.135	**5.502**	**6.715**

Table 11: Performance comparison of few-shot scenarios on METR-LA dataset at different prediction steps in terms of MAE and RMSE. Bold denotes the best results and underline denotes the second-best results.

	MAE			RMSE		
Model	**Step 1**	**Step 3**	**Step 6**	**Step 1**	**Step 3**	**Step 6**
HA	3.142	3.142	3.142	4.535	4.535	4.535
ARIMA	5.179	5.456	5.413	6.829	7.147	7.144
RegionTrans	2.388	2.845	3.071	3.464	4.166	4.488
DASTNet	2.278	2.818	3.164	3.256	3.909	4.427
MAML	2.396	2.885	3.181	3.327	4.050	4.499
TPB	2.156	2.593	3.116	<u>3.082</u>	<u>3.637</u>	4.382
STGFSL	<u>2.133</u>	<u>2.565</u>	<u>2.901</u>	3.183	3.815	<u>4.291</u>
Ours	**2.031**	**2.381**	**2.550**	**2.948**	**3.422**	**3.630**

Table 12: Performance comparison of few-shot scenarios on Didi-Chengdu dataset at different prediction steps in terms of MAE and RMSE. Bold denotes the best results and <u>underline</u> denotes the second-best results.