Cross-city Few-Shot Traffic Forecasting via Traffic Pattern Bank

Zhanyu Liu
Shanghai Jiao Tong University
Shanghai, China
zhyliu00@sjtu.edu.cn

Guanjie Zheng*
Shanghai Jiao Tong University
Shanghai, China
gjzheng@sjtu.edu.cn

Yanwei Yu
Ocean University of China
Qingdao, China
yuyanwei@ouc.edu.cn

ABSTRACT

Traffic forecasting is a critical service in Intelligent Transportation Systems (ITS). Utilizing deep models to tackle this task relies heavily on data from traffic sensors or vehicle devices, while some cities might lack device support and thus have few available data. So, it is necessary to learn from data-rich cities and transfer the knowledge to data-scarce cities in order to improve the performance of traffic forecasting. To address this problem, we propose a cross-city few-shot traffic forecasting framework via **T**raffic **P**attern **B**ank (TPB) due to that the traffic patterns are similar across cities. TPB utilizes a pre-trained traffic patch encoder to project raw traffic data from data-rich cities into high-dimensional space, from which a traffic pattern bank is generated through clustering. Then, the traffic data of the data-scarce city could query the traffic pattern bank and explicit relations between them are constructed. The metaknowledge is aggregated based on these relations and an adjacency matrix is constructed to guide a downstream spatial-temporal model in forecasting future traffic. The frequently used meta-training framework *Reptile* is adapted to find a better initial parameter for the learnable modules. Experiments on real-world traffic datasets show that TPB outperforms existing methods and demonstrates the effectiveness of our approach in cross-city few-shot traffic forecasting.

CCS CONCEPTS

• **Information systems** → **Spatial-temporal systems**.

KEYWORDS

Traffic Forecasting, Few-shot learning, Spatial-temporal data

ACM Reference Format:
Zhanyu Liu, Guanjie Zheng, and Yanwei Yu. 2023. Cross-city Few-Shot Traffic Forecasting via Traffic Pattern Bank. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3614829

1 INTRODUCTION

Traffic forecasting has been a critical service in Intelligent Transportation Systems (ITS). The data used in traffic forecasting is mostly from traffic sensors [24] or vehicle device [7]. When devices are sparse and the collected data becomes scarce, it will be challenging to train a deep model. Consequently, learning from the data-rich cities and transferring the knowledge to the data-scarce city could be a promising solution to the cross-city few-shot traffic forecasting problem.

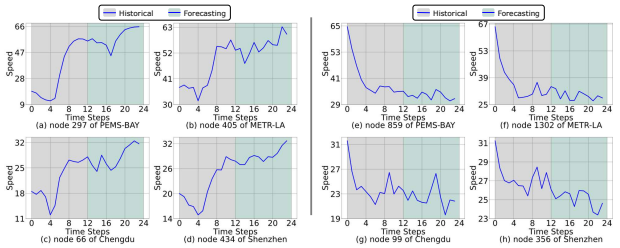

Figure 1: Examples of the observation in traffic forecasting that similar historical traffic (gray part) leads to similar future traffic (green part). (a)~(d) The historical speed shows a rapid increase, and the future speed continues to increase. (e)~(h) The historical speed has just fallen to a low value, and the future speed fluctuates and decreases slightly.

Currently, many studies have been working on this problem. RegionTrans [41] and CrossTRes [18] work on grid-based city data and aim to use advanced techniques to learn the region correlation between the data-rich cities and data-scarce city to transfer the knowledge better. MetaST [46] uses a global learnable memory to store the knowledge. However, these methods utilize auxiliary data such as event information to transfer knowledge, which is incompatible with the city with little similar auxiliary information. ST-MetaNet [34] and ST-GFSL [29] utilize metaknowledge to generate the parameters of spatial-temporal neural networks and use the network to forecast future traffic data. However, these methods learn the metaknowledge implicitly while ignoring the strong and explicit correlation between the traffic data of different cities and thus become less effective.

In fact, the traffic pattern is similar across cities. Namely, when the historical traffic on roads of several cities is similar, their future traffic tends to be similar. Fig. 1 shows some examples of the fact on four traffic datasets. In this figure, (a)~(d) show that a rapid increase of speed manifests the future speed will continue to increase, and (e)~(h) show a rapid decrease of speed manifests the future speed will fluctuate and decrease slightly. If these traffic patterns could be captured, the model could explicitly relate the present traffic in the data-scarce city to the traffic patterns of data-rich cities and precisely forecast future traffic.

Inspired by this fact, we propose a novel cross-city few-shot traffic forecasting framework via **T**raffic **P**attern **B**ank, which is

*corresponding author

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3614829

abbreviated as **TPB**. Specifically, we first pre-train a traffic patch encoder in data-rich cities. The pre-trained encoder captures the general dynamics of traffic and projects the raw traffic data to a high dimensional space. Furthermore, we input a large number of raw traffic data of data-rich cities into the pre-trained encoder to get traffic patch embeddings. A clustering algorithm is then applied to the enormous traffic patch embeddings to eliminate redundancy and a traffic pattern bank is generated. The traffic pattern bank contains the metaknowledge that helps forecast future traffic accurately. Finally, the input traffic data of the data-scarce city queries the traffic pattern bank and explicit relations are constructed. The metaknowledge is aggregated based on the relations and an adjacency matrix is constructed based on the metaknowledge to guide the downstream spatial-temporal model in forecasting future traffic. The *Reptile* meta-learning framework [30] is utilized to find a better initial parameter of the learnable modules. Experiments show that TPB achieves an average performance enhancement of 8.4% and 11.3% compared with the baselines in RMSE and MAPE respectively.

In summary, the main contributions of our work are as follows.

- We investigate the cross-city few-shot traffic forecasting task and demonstrate that similar traffic patterns across cities are helpful to forecast future traffic. This explicit relation between data-rich cities and the data-scarce city is ignored by previous research.
- We propose a novel cross-city few-shot traffic forecasting framework TPB. The framework effectively leverages the strong and explicit correlations between traffic patterns in different cities to better forecast future traffic in the data-scarce city.
- We demonstrate the effectiveness of the TPB framework through extensive experiments on real-world traffic datasets. The results show that TPB is able to achieve superior performance compared to state-of-the-art methods for cross-city few-shot traffic forecasting.

2 RELATED WORK

Traffic Forecasting Traffic Forecasting has been a hot research area for its important application. Conventional traffic forecasting methods that utilize Kalman Filter [26, 32], SVM [31], Bayesian Network [51], probalistic model [1], or simulation [25] get good results. Recently the prosperity of GCN [19], RNN [15], Attention [40] contributes to the deep methods for modeling the spatial-temporal traffic graph thus benefiting the area of traffic forecasting. DCRNN [24], STGCN [48], STDN [47], GSTNet [9], LSGCN [16], STFGNN [23] combine modules such as GCN, GRU, and LSTM to model the spatial-temporal relation. To better capture the dynamic spatial-temporal relations of the nodes of the traffic graph, methodologies such as AGCRN [2], Graph Wavenet [45], GMAN [50], FC-GAGA [33], D2STGNN [38], HGCN [12], ST-WA [5], DSTAGNN [20] utilize techniques such as attention to reconstruct the adaptive adjacent matrix and fuse the temporal long term relation to make better predictions. MTGNN [44] and DMSTGCN [13] utilize auxiliary information that helps forecast the traffic. STGODE [10], STG-NCDE [4], STDEN [17] model the traffic based on ordinary differential equation(ODE). DGCNN [6], StemGNN [3] view the traffic forecasting task in a graph spectral view. PM-MemNet [21] learns and clusters the traffic flow patterns. STEP [37] adapts MAE [14] and proposes a pipeline to pre-train a model. FDTI [28] builds layer graphs to conduct fine-grained traffic forecasting. However, these methods focus on single-city traffic forecasting. When facing cross-city few-shot traffic forecasting, some of their modules do not work(such as node embedding) and achieve poor results.

Cross-City Few-Shot Learning Recently Few-Shot Learning has yielded satisfying results in many areas such as computer vision [39], natural language processing [22], and reinforcement learning [11] when facing the problem of data scarcity. In the area of urban computing, some methods aim to solve city data scarcity by transferring city knowledge. Floral [42] transfers the knowledge of rich multimodal data and labels to the target city to conduct air quality prediction, which is mainly designed for classification problems. RegionTrans [41] and CrossTReS [18] learn the region correlation between the source cities to the target city. MetaST [46] learns a global memory which is then queried by the target region. STrans-GAN [49] utilizes the GAN-based model to generate future traffic based on traffic demand. However, these methods focus on grid-based multimodal city region data rather than graph-based city data. ST-MetaNet [34] and ST-GFSL [29] generate the parameters of spatial-temporal neural networks according to the learned meta-knowledge. However, these methods essentially propose another way to initialize the parameter without utilizing the rich information and strong correlation between the traffic patterns of cities.

3 PRELIMINARY

Traffic Spatio-Temporal Graph: A traffic spatio-temporal graph can be denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$. $\mathcal{V}$ is the set of nodes and $N = |\mathcal{V}|$ is the number of nodes. $\mathcal{E}$ is the set of edges and each edge can be denoted as $e_{ij} = (v_i, v_j)$. $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix of $\mathcal{G}$, where $a_{ij} = 1$ indicates there is an edge between v_i and v_j. By denoting T_{total} as the total number of time steps, $\mathbf{X} \in \mathbb{R}^{N \times T_{total} \times C}$ represents the node feature that contains the traffic time series data, such as traffic speed, traffic volume, and time of day. Here, C indicates the input channel and $\mathbf{X}_t \in \mathbb{R}^{N \times C}$ represents the traffic data at time step t.

Traffic Patch & Pattern: Traffic patch is a fixed interval traffic time series data. For example, given traffic time series data of one day, we can split the data to get 24 one-hour traffic patches. Formally, given traffic time series data of v_i as $\mathbf{X}^i$, one traffic patch is $\mathbf{S}^i_t := \mathbf{X}^i_{t:t+T_0} \in \mathbb{R}^{T_0 \times C}$. In this paper, each patch contains 12 points, i.e., $T_0 = 12$. Given enormous traffic patches of raw input data, the traffic patterns could be refined from them.

Traffic Forecasting: Given the historical traffic data of T steps, the goal of the traffic forecasting problem is to learn a function $f(\cdot)$ that forecasts the future traffic data of T' steps. This task is formulated as follows.

$$[\mathbf{X}^{t-T+1}, \cdots, \mathbf{X}^t] \xrightarrow{f(\cdot)} [\mathbf{X}^{t+1}, \cdots, \mathbf{X}^{t+T'}] \tag{1}$$

Cross-city Few-Shot Traffic Forecasting: Given P source cities $\mathcal{G}^{source} = \{\mathcal{G}^{source}_1, \cdots, \mathcal{G}^{source}_P\}$ with a large amount of traffic data and a target city $\mathcal{G}^{target}$ with only a few traffic data, the goal of cross-city few-shot traffic forecasting is to learn a model based

on the available data of both $\mathcal{G}^{source}$ and $\mathcal{G}^{target}$ to conduct traffic forecasting on the future data of $\mathcal{G}^{target}$.

4 METHODOLOGY

In this section, we introduce our framework **TBP** to tackle the few-shot traffic forecasting task. First, a traffic patch encoder is pre-trained by the data of source cities in the fashion of the Masked Autoencoder [14, 37]. Then, the traffic pattern bank is constructed by clustering the high-dimensional traffic patch encoder space. Finally, the metaknowledge is aggregated based on the traffic pattern bank and a metaknowledge-based adjacency matrix is constructed to guide a downstream spatial-temporal model to forecast future traffic. The whole diagram is shown in Fig. 2

4.1 Traffic Patch Encoder Pre-training

***Goal*:** The input raw traffic patch data of source cities contain rich information which could serve as the metaknowledge for the target city to conduct traffic few-shot forecasting. However, the raw traffic patch is contaminated by the noise in the collecting, aggregating, splitting process, and so on. Furthermore, the raw traffic patch lies in low-dimension space, which lacks expressiveness for downstream pattern generation. So, the goal of this stage is to train a traffic patch encoder that projects the input traffic patches to a higher-dimensional denoised embedding space and thus generates better traffic patch embeddings for the downstream stages. Here, we propose a traffic patch encoder pre-training framework based on SOTA methods STEP [37] and MAE [14]. Other pre-training frameworks could also be adapted here.

Masked Input: The input is P continuous traffic patches of v_i, denoted as $\mathbf{S}^i = \{\mathbf{S}^i_1, \cdots, \mathbf{S}^i_P\}$, where the shape of each patch is $\mathbf{S}^i_j \in \mathbb{R}^{T_0 \times C}$. Here, C is the input channel, T_0 is set to 12, and P is set to 24. Note that the interval between data points is 5 minutes so the actual length of each patch is one hour and the actual length of historical patches is one day. We randomly set a subset of the traffic patches and mask them. The selected patches are not required to be continuous. We set the mask ratio to 75% according to the original papers [14, 37]. Such a high mask ratio makes the pre-training task difficult so the pre-trained encoder is expected to be more robust and potentially more effective when transferred to other cities.

Encoder: The encoder should be able to aggregate the information on the remaining parts of the traffic patches. The models based on RNN are not suitable here since most of the patches are masked. Here, we adopt transformer [40] as the encoder of the masked traffic patches. Before being put into the transformer layers, each unmasked raw patch $\mathbf{S}^i_j$ is fed into a linear layer. Then the learnable positional embedding based on the time of the week is added. There is a total of 168 learnable positional embeddings since there are 24×7 hours in a week. After adding the positional embedding, the unmasked patches are put into the transformer encoder and the corresponding output embeddings are generated. Formally, we can write as follows.

$$\mathbf{H}^i_j = TS_E(Concat\{(\mathbf{W}_{enc}\mathbf{S}^i_j + \mathbf{b}_{enc}) + \mathrm{PE}(\mathbf{S}^i_j) | \mathrm{M}^i_j = 0\}) \quad (2)$$

Here, PE() indicates learnable positional embedding and $TS_E()$ indicates the encoder transformer layers. $\mathrm{M}^i_j = 0$ indicates $\mathbf{S}^i_j$ is not masked and vice versa. $\mathbf{W}_{enc} \in \mathbb{R}^{d \times T_0 C}$ and $\mathbf{b}_{enc} \in \mathbb{R}^d$ are learnable parameters.

Algorithm 1: Traffic Pattern Generation

Input: Pre-trained traffic patch encoder TS_E, Historical traffic patch of source cities $\mathbf{S}^i_j$, number of clusters K.
Output: Traffic pattern bank $\mathbf{B} \in \mathbb{R}^{K \times d}$

```
for i ← range(0, N, 1) do
    for j ← range(0, T_total, P) do
        H^i_j, ··· , H^i_{j+P} = TS_E(S^i_j, ··· , S^i_{j+P});
    end
end
Collect {H^i_j | i = 1, ..., N; j = 0, ..., T_total};
o_{1:K} = KMeans({H^i_j});
B ← o_{1:K};
return B;
```

Decoder: The decoder takes in the unmasked patch embeddings to reconstruct the whole time series back to its original numerical values. First, a learnable masked token $\mathbf{H}_{mt} \in \mathbb{R}^d$ is duplicated and fills the masked positions. After filling the masked positions, the number of patches aligns with the original input. Then, the filled patch series is put into the transformer layer to aggregate the patch series information. Finally, the output embedding of the transformer layer is fed into a linear layer to reconstruct the original time series, noted as $\hat{\mathbf{S}}^i_j$. Formally, the decoder process could be represented as follows.

$$\hat{\mathbf{S}}^i_j = \mathbf{W}_{dec} \cdot TS_D(Concat\{\mathbf{H}^i_j, \mathbf{H}_{mt} | \mathrm{M}^i_j = 0\}) + \mathbf{b}_{dec} \quad (3)$$

Here, $TS_D()$ indicates the decoder transformer layer, and $\mathbf{W}_{dec} \in \mathbb{R}^{T_0 C \times d}$ and $\mathbf{b}_{dec} \in \mathbb{R}^{T_0 C}$ are learnable parameters.

Loss Module: The task of the pre-training is to reconstruct the masked traffic patches while the unmasked patch is the input of the model. So, the final loss only contains the reconstruction Mean Square Error (MSE) of masked traffic patches. Formally, the loss of the pre-training stage is as follows.

$$\mathcal{L}_p = \sum_{i=1}^{N} \sum_{j=1}^{P} \mathrm{M}^i_j (\mathbf{S}^i_j - \hat{\mathbf{S}}^i_j)^2 \quad (4)$$

In summary, the pre-training stage reconstructs the mask traffic patch. During the pre-training, the encoder is able to learn useful features about the traffic patch and project the traffic patches to a higher-dimensional denoised embedding space, which benefits the process of pattern generation.

4.2 Traffic Pattern Generation

In this part, we propose a framework that generates the representative traffic patch embeddings, i.e., traffic patterns. The framework is based on the patch encoder pre-trained in the previous stage. The algorithm is shown in Alg. 1.

***Goal*:** The enormous traffic patches in the source cities contain rich information, which could serve as the metaknowledge for the few-shot traffic forecasting on the target city. However, directly using the enormous traffic patches to make predictions may not

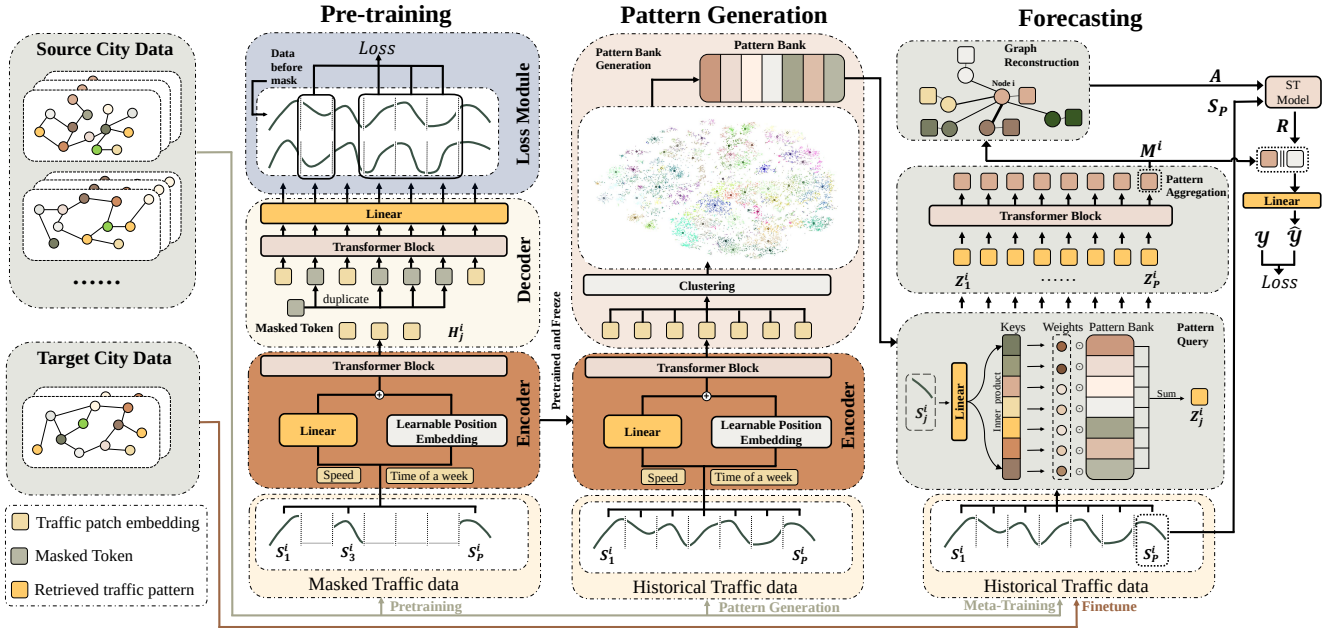

Figure 2: Diagrams of TPB. 1) In the pre-training stage, a traffic patch encoder is pre-trained by the data of source cities. 2) In the pattern generation stage, the traffic patches of source cities are put into the pre-trained encoder and the output patch embeddings are clustered to get the traffic pattern bank. 3) In the forecasting stage, the traffic pattern bank is queried by the input traffic patches and the acquired metaknowledge helps the downstream spatial-temporal model (STmodel) make accurate predictions.

be practical since the number of patches is too massive. So, we encode the traffic patches and use clustering techniques to identify representative traffic patches to construct a small traffic pattern bank. The small traffic pattern bank not only reduces the computational burden of the forecasting model but also serves as a compact representation of the traffic patterns in the source cities.

Framework: We directly utilize the traffic patch encoder $TS_E()$ that is pre-trained in the previous stage. The enormous traffic patches in the source cities $\mathbf{S}_j^i$ are put into the patch encoder to generate traffic patch embeddings $\mathbf{H}_j^i$.

$$\mathbf{H}_0^i, \mathbf{H}_1^i, \cdots, \mathbf{H}_P^i = TS_E(\mathbf{S}_0^i, \mathbf{S}_1^i, \cdots, \mathbf{S}_P^i) \tag{5}$$

The pre-trained patch encoder could project the traffic patches into high-dimensional embedding space, and traffic patches with similar traffic semantics are projected to near space.

After the traffic patches are projected into the embedding space, we can then use clustering techniques to group similar traffic patches together. Several different approaches can be taken when it comes to clustering the traffic patches, including unsupervised techniques like k-means and density-based clustering. Here, to simplify the framework and get robust traffic patterns, we adopt *k-means* as the clustering algorithm and *cosine* as the distance function.

$$o_{1:K} = KMeans(Concat\{\mathbf{H}_j^i | i = 1, ..., N; j = 0, ..., T_{total}\}) \tag{6}$$

Here, $o_{1:K} \in \mathbb{R}^{K \times d}$ are the centroids of the clusters.

Once the traffic patches have been grouped into clusters, we can then select a representative set of traffic patterns from each cluster to construct the traffic pattern bank. This can be done in a variety of ways, such as selecting the mean embedding in each cluster, selecting the patch that is nearest to the centroid, or using some other metric to determine the most representative traffic pattern. In the few-shot traffic forecasting setting, the traffic patches of the target city are unknown and some distribution bias could exist. So, the centroid represents the general characteristics of the traffic patterns in the cluster rather than being specific to any particular traffic patch. We choose the centroids of each cluster $o_{1:K}$ as the traffic pattern bank $\mathbf{B}$ to guarantee robustness and transferability. The traffic pattern bank will offer the metaknowledge in the few-shot traffic forecasting stage.

4.3 Forecasting Stage

In this part, we propose a framework that utilizes the metaknowledge from the traffic pattern bank learned in the data of source cities to conduct few-shot traffic forecasting. The core idea is to query the traffic pattern bank with historical traffic patches and aggregate the retrieved patterns as the auxiliary metaknowledge. The metaknowledge is used to reconstruct graph structure and guide the downstream spatial-temporal model to forecast future traffic.

***Goal*:** The goal of this framework is to use the metaknowledge from the traffic pattern bank to improve the accuracy of few-shot traffic forecasting in the target city.

Pattern Query: The traffic pattern bank consists of two components: *Key* and Pattern Bank $\mathbf{B}$. The *Key* is a learnable embedding matrix with shape $\mathbb{R}^{K \times d_q}$, where K is the number of patterns and

Algorithm 2: Meta-training and Fine-tuning Process

Input: Forecasting model $F_\theta(\cdot)$, Source city data G_{source}, Target city data G_{target}
Output: Trained Model Parameter θ

/* Meta-training the θ with G_{source} */
Random initialize θ;
for $e \leftarrow range(0, meta\_epochs, 1)$ **do**
  // sample support and query tasks τ_{spt} and τ_{qry}
  $\tau_{spt}, \tau_{qry} \longleftarrow SampleTask(G_{source})$;
  // get historical data S and label data $\mathcal{Y}$
  $\mathbf{S}_{spt}, \mathcal{Y}_{spt} \longleftarrow \tau_{spt}$;
  $\mathbf{S}_{qry}, \mathcal{Y}_{qry} \longleftarrow \tau_{qry}$;
  $\theta_\tau \longleftarrow \theta$;
  // compute the gradient of each step
  for $i \leftarrow range(0, update\_step, 1)$ **do**
    $\hat{\mathcal{Y}}^{\theta_\tau}_{spt} \longleftarrow F_{\theta_\tau}(S_{spt})$;
    compute $\nabla_{\theta_\tau} \mathcal{L}_i(\hat{\mathcal{Y}}^{\theta_\tau}_{spt}, \mathcal{Y}_{spt})$ by Eq (14);
    $\theta_\tau \longleftarrow \theta_\tau - \alpha \nabla_{\theta_\tau} \mathcal{L}_i(\hat{\mathcal{Y}}^{\theta_\tau}_{spt}, \mathcal{Y}_{spt})$;
    $\hat{\mathcal{Y}}^{\theta_\tau}_{qry} \longleftarrow F_{\theta_\tau}(S_{qry})$;
    compute & store $\nabla_{\theta_\tau} \mathcal{L}_i(\hat{\mathcal{Y}}^{\theta_\tau}_{qry}, \mathcal{Y}_{qry})$;
  end
  // use the gradients to update θ
  for $i \leftarrow range(0, update\_step, 1)$ **do**
    $\theta \longleftarrow \theta - \frac{\beta}{update\_step} \nabla_{\theta_\tau} \mathcal{L}_i(\hat{\mathcal{Y}}^{\theta_\tau}_{qry}, \mathcal{Y}_{qry})$
  end
end
/* Finetune the θ with G_{target} */
for $e \leftarrow range(0, train\_epochs, 1)$ **do**
  for *Iterate All Batches* **do**
    $\mathbf{S}, \mathcal{Y} \longleftarrow SampleBatch(G_{target})$;
    $\hat{\mathcal{Y}}^{\theta} \longleftarrow F_\theta(\mathbf{S})$;
    compute $\nabla_\theta \mathcal{L}(\hat{\mathcal{Y}}^{\theta}, \mathcal{Y})$ by Eq (14);
    use $\nabla_\theta \mathcal{L}(\hat{\mathcal{Y}}^{\theta}, \mathcal{Y})$ to update θ with Adam optimizer;
  end
end
return θ

d_q is the embedding shape of *Key*. The Pattern Bank $\mathbf{B} \in \mathbb{R}^{K \times d}$ is a fixed embedding matrix learned from the previous clustering stage. Suppose the P continuous traffic patches of v_i are denoted as $\mathbf{S}^i = \{\mathbf{S}^i_1, \cdots, \mathbf{S}^i_P\}$. For each patch $\mathbf{S}^i_j$, it will be used as the query to the traffic pattern bank. To do this, we first compute the dot product scores ω between the linear transformed query patch and each key in the *Key* matrix.

$$\omega^k = dot(\mathbf{W}_q \mathbf{S}^i_j + \mathbf{b}_q, Key^k) \tag{7}$$

Here, $\mathbf{W}_q \in \mathbb{R}^{d_q \times d}$ and $\mathbf{b}_q \in \mathbb{R}^{d_q}$ represents the linear layer. $Key^k \in \mathbb{R}^{d_q}$ is the *k-th* key of the traffic pattern and $\omega^k \in \mathbb{R}$ is the score. Then, the retrieved traffic pattern $\mathbf{Z}^i_j \in \mathbb{R}^d$ could be represented as a weighted sum of traffic pattern $\mathbf{B}^k \in \mathbb{R}^d$ as follows.

$$\mathbf{Z}^i_j = \sum_{k=1}^{K} \omega^k \cdot \mathbf{B}^k \tag{8}$$

This process allows the model to retrieve the most relevant traffic patterns for a given query patch, which can then be used for further metaknowledge generation.

Pattern Aggregation: Once the retrieved traffic pattern $\mathbf{Z}^i_j$ has been identified for each patch $\mathbf{S}^i_j$, the metaknowledge $\mathbf{M}^i$ of the historical traffic data is extracted from the retrieved traffic pattern series. We use transformer layers to achieve this goal. Formally, by denoting the retrieved traffic pattern series as $\{\mathbf{Z}^i_1, ..., \mathbf{Z}^i_P\}$, the metaknowledge $\mathbf{M}^i \in \mathbb{R}^k$ is represented as the last hidden states of the output of the transformer layer, which is shown as follows.

$$\mathbf{M}^i = Last(TS(\{\mathbf{Z}^i_1, ..., \mathbf{Z}^i_P\})) \tag{9}$$

Here, *TS*() is the transformer layer and *Last*() takes the last hidden states of the output of the transformer.

Graph Reconstruction: To better describe the correlations between the nodes in the target city, we reconstruct an adaptive adjacency matrix based on the metaknowledge $\mathbf{M}^i$ of each node. Here we did not use the original prior-knowledge-based graph since the adaptive graph is more effective in the traffic forecasting scenarios [27]. The reconstructed adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ is a weighted representation of the connectivity between nodes in the graph, where N is the number of nodes in the graph. To reconstruct the adjacency matrix, we first linearly project the metaknowledge $\mathbf{M}^i$ to query space and key space.

$$\begin{aligned} \mathbf{Q}^i &= \mathbf{W}_Q \mathbf{M}^i + \mathbf{b}_Q \\ \mathbf{K}^i &= \mathbf{W}_K \mathbf{M}^i + \mathbf{b}_K \end{aligned} \tag{10}$$

Here, $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_Q, \mathbf{b}_K \in \mathbb{R}^d$ are shared learnable matrix. Then we compute the dot product scores r between the metaknowledge representations in query space and key space for each pair of nodes:

$$r_{i,j} = dot(\mathbf{Q}^i, \mathbf{K}^j) \tag{11}$$

These scores can then be used to compute the entries of the reconstructed adjacency matrix by using softmax with temperature ϵ as follows.

$$\mathbf{A}_{i,j} = \frac{exp(r_{i,j}/\epsilon)}{\sum_{k=1}^{N} exp(r_{i,k}/\epsilon)} \tag{12}$$

The reconstructed adjacency matrix encodes the relative importance of the metaknowledge connections between nodes in the traffic graph, with higher values indicating stronger connections. By using the metaknowledge representations of the nodes, the model is able to capture the more complex and nuanced relationships in the few-shot setting.

Metaknowledge-auxiliary Forecasting: After reconstructing the adjacency matrix, we can use it to guide the downstream spatial-temporal model (STmodel) in forecasting future traffic. A normal STmodel takes in near traffic patches $\mathbf{S}_P \in \mathbb{R}^{N \times T_0 \times C}$ and the adjacency matrix $\mathbf{A}$ and extracts the traffic information to generate the representations $\mathbf{R} \in \mathbb{R}^{N \times d}$. The TBP framework could extend to almost every STmodel and we choose Graph Wavenet [45] as

our backend due to its simplicity and effectiveness. Here, the metaknowledge is fused into the adjacency matrix and helps the STmodel capture the dependencies between different nodes. To further incorporate the metaknowledge into the forecasting process, we concatenate the metaknowledge $\mathbf{M} \in \mathbb{R}^{N\times d}$ with the representation $\mathbf{R} \in \mathbb{R}^{N\times d}$. The resulting representation is then fed into a regression layer to generate the forecast traffic data $\hat{\mathcal{Y}}$, which is a Multi-Layer Perception (MLP). This process can be represented as follows.

$$\begin{aligned}\hat{\mathcal{Y}} &= MLP([\mathbf{M}||\mathbf{R}]) \\ &= MLP([\mathbf{M}||STModel(\mathbf{S}_P, \mathbf{A})])\end{aligned} \tag{13}$$

Here, $\hat{\mathcal{Y}} \in \mathbb{R}^{N\times T'\times C}$ is the forecasting result. Given the ground truth $\mathcal{Y} \in \mathbb{R}^{N\times T'\times C}$, mean square error is selected as the loss:

$$\mathcal{L} = \frac{1}{NT'C}\sum_{i=1}^{N}\sum_{j=1}^{T'}\sum_{k=1}^{C}(\mathcal{Y}_{ijk} - \hat{\mathcal{Y}}_{ijk})^2 \tag{14}$$

Meta-Training & Fine-tuning: Though the traffic pattern bank contains rich metaknowledge about the traffic patches of source cities, there are other models that need to adapt to the target city such as transformer layer in *Pattern Aggregation*, $\mathbf{Q}, \mathbf{K}$ matrices in *Graph Reconstruction*, and STmodel in *Metaknowledge-auxiliary Forecasting*. These learnable models are collectively referred to as $F_\theta(\cdot)$. To address this issue, we propose a meta-learning based approach to learn better initial parameters for these models from rich the source city data G_{source}. Then, we fine-tune the model with the few-shot data of the target city G_{target}. We adopt *Reptile* [30] meta-learning framework here, which essentially conducts multi-step gradient descent on the query tasks. Formally, by denoting the forecasting model as $F_\theta(\cdot)$ and its corresponding parameter as θ, the meta-training and fine-tuning process are shown in Alg. 2.

5 EXPERIMENT

In this section, we evaluate our proposed framework TPB in various aspects through extensive experiments. Specifically, the following research questions are answered.

- **RQ1:** How does TPB perform against other baselines in the few-shot traffic forecast task?
- **RQ2:** Is TPB able to improve the performance of different STmodels on few-shot forecasting?
- **RQ3:** How to choose the clustering parameter K to get a good traffic pattern bank?
- **RQ4:** How does each component of TPB contribute to the final forecasting performance?

5.1 Experiment Settings

Dataset: We evaluate our proposed framework on four commonly used real-world public datasets: *PEMS-BAY*, *METR-LA* [24], *Chengdu*, *Shenzhen* [7]. These datasets contain months of traffic speed data. The statistical details of these data are listed in Table 1.

Few-shot Setting: We use a similar few-shot traffic forecasting setting to [29]. The data of these four cities are divided into source data, target data, and test data. Here, source data consists of data from three cities, while target and test data consist of data from the target city. For example, if *PEMS-BAY* is selected as the target city, the full data of *METR-LA*, *Chengdu*, *Shenzhen* constitutes source data. Then, two-day data of *PEMS-BAY* constitutes the target data and the remaining data of *PEMS-BAY* constitutes the test data. The Pre-training and Pattern Generation are conducted in the source data, and the Meta-training&Fine-tuning Process are conducted in the target data. Finally, we evaluate our framework in the test data.

Table 1: Statistical details of traffic datasets.

	PEMS-BAY	METR-LA	Chengdu	Shenzhen
# of Nodes	325	207	524	627
# of Edges	2,694	1,722	1,120	4,845
Interval	5 min	5 min	10 min	10 min
# of Time Step	52,116	34,272	17,280	17,280
Mean	61.7768	58.2749	29.0235	31.0092
Std	9.2852	13.1280	9.6620	10.9694

Implementation: In Pre-training and Pattern Generation phases, we use $T_0 = 12$ and $P = 24$, which means one-day data is divided into 24 patches to form a patch series. In the Forecasting stage, we also use the same T_0 and P to forecast the future 6 steps of data. Besides, *Chengdu* and *Shenzhen* have a longer data interval so we conduct linear interpolate to align the positional embedding. Only speed is contained in the dataset and thus the data channel C is 1. The mask ratio of Pre-training is set to 75% according to [37]. The learning rate of Pre-training is set to 0.0001 and the learning rate of Meta-training α and β are both set to 0.0005. The Adam optimizer of the Fine-tuning has a learning rate of 0.001 and weight decay of 0.01. The dimension of H and Key and the positional embedding size are set to 128. The batch size is set to 4. The number of tasks of meta-training is set to 2. The experiment is implemented by Pytorch 1.10.0 on RTX3090. The code is in https://github.com/zhyliu00/TPB.

5.2 RQ1: Overall Performance

Baselines: We select 10 baselines of four types to evaluate the performance of TPB on the few-shot forecasting task. These baselines include traditional methods, typical deep-learning traffic forecasting methods, pre-training traffic forecasting methods, and cross-city traffic forecasting methods. We use the model structure hyperparameters reported by the original papers of these models or frameworks. Note that the typical deep-learning traffic forecasting methods are implemented in *Reptile*, which is a meta-learning framework. We guarantee the fairness of the comparisons of different baselines. For the hyper-parameter of meta-learning framework, we hold the same sufficient strategy for meta-learning hyperparameters searching in all methods including TPB to guarantee fairness. The learning rate of meta-training & fine-tuning is set to 5e-4, the meta-training epoch is searched in range(5, 30, 5), the fine-tuning epoch is searched in range(50, 400, 50), update_step is searched in range(2, 5, 1) (larger update_step would lead to GPU memory limit exceeded).

- Traditional methods
  - **HA**: Historical Average, which uses the average of previous periods as the predictions of future traffic.
  - **ARIMA** [43]: A commonly used statistical time series forecasting model.
- Typical deep-learning traffic forecasting methods (*Reptile*)

Table 2: Overall performance of few-shot traffic forecasting on *PEMS-BAY*, *METR-LA*, *Chengdu*, and *Shenzhen*. M,C,S→PEMS-BAY means the source data is METR-LA, Chengdu, Shenzhen and the target data is PEMS-BAY. The mean and standard deviation of the results in 5 runs is shown. In each column, the best result is highlighted in bold and grey, and the second-best result is underlined. Marker * and ** indicates the mean of the results is statistically significant (* means t-test with p-value < 0.05 and ** means t-test with p-value < 0.01).

	M,C,S→PEMS-BAY						P,C,S→METR-LA					
	5 min		15 min		30 min		5 min		15 min		30 min	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
HA	5.49	2.67	6.01	2.95	6.56	3.23	7.37	3.97	8.05	4.27	8.76	4.66
ARIMA	4.12	2.02	4.98	2.35	5.14	2.50	4.79	3.04	6.27	3.48	7.54	4.31
DCRNN	2.38 (0.04)	1.45 (0.02)	3.41 (0.07)	1.90 (0.03)	4.59 (0.10)	2.40 (0.06)	4.76 (0.08)	2.96 (0.05)	6.11 (0.09)	3.45 (0.07)	7.81 (0.12)	4.30 (0.06)
GWN	2.41 (0.18)	1.46 (0.10)	3.52 (0.14)	1.96 (0.06)	4.67 (0.10)	2.36 (0.08)	4.28 (0.11)	2.68 (0.17)	5.72 (0.17)	3.30 (0.15)	7.39 (0.26)	4.11 (0.20)
STFGNN	2.31 (0.14)	1.30 (0.04)	3.42 (0.07)	1.81 (0.03)	4.56 (0.05)	2.34 (0.07)	4.39 (0.10)	2.61 (0.07)	5.76 (0.07)	3.17 (0.10)	7.10 (0.08)	3.91 (0.11)
DSTAGNN	2.19 (0.17)	1.32 (0.07)	3.40 (0.15)	1.89 (0.14)	4.87 (0.26)	2.70 (0.33)	4.35 (0.08)	2.59 (0.05)	5.84 (0.16)	3.38 (0.08)	7.68 (0.43)	4.36 (0.34)
FOGS	2.33 (0.23)	1.36 (0.18)	3.43 (0.21)	1.92 (0.16)	4.53 (0.15)	2.38 (0.09)	4.34 (0.12)	2.56 (0.78)	6.11 (0.14)	3.36 (0.10)	7.40 (0.20)	3.99 (0.16)
STEP	2.15 (0.04)	1.38 (0.02)	3.26 (0.05)	1.79 (0.04)	4.33 (0.08)	2.32 (0.04)	4.26 (0.06)	2.60 (0.07)	5.63 (0.09)	3.26 (0.09)	7.18 (0.15)	3.98 (0.12)
AdaRNN	1.98 (0.04)	1.18 (0.03)	3.30 (0.05)	1.75 (0.04)	4.40 (0.08)	2.38 (0.03)	4.41 (0.14)	2.60 (0.10)	5.77 (0.15)	3.18 (0.10)	7.33 (0.18)	3.90 (0.17)
ST-GFSL	2.01 (0.08)	1.18 (0.05)	3.19 (0.07)	1.73 (0.07)	4.57 (0.10)	2.22 (0.06)	4.23 (0.09)	2.43 (0.08)	5.72 (0.12)	3.03 (0.09)	7.28 (0.21)	3.87 (0.15)
TPB	**1.88 (0.02)****	**1.07 (0.02)****	**3.13 (0.04)****	**1.57 (0.02)****	**4.27 (0.06)****	**2.06 (0.03)****	**4.13 (0.07)****	**2.39 (0.06)****	**5.55 (0.09)****	**2.90 (0.07)****	**6.91 (0.12)****	**3.69 (0.08)****
	P,M,S→Chengdu						P,M,C→Shenzhen					
	10 min		30 min		60 min		10 min		30 min		60 min	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
HA	4.29	3.02	4.80	3.42	5.44	3.91	3.81	2.55	4.21	2.82	4.73	3.19
ARIMA	3.79	2.75	4.40	3.21	5.09	3.60	3.38	2.34	4.10	2.78	4.68	3.08
DCRNN	3.27 (0.07)	2.28 (0.06)	4.13 (0.07)	2.89 (0.07)	4.78 (0.10)	3.39 (0.08)	2.81 (0.05)	1.94 (0.04)	3.59 (0.07)	2.41 (0.06)	4.11 (0.08)	2.78 (0.05)
GWN	3.33 (0.15)	2.34 (0.14)	4.22 (0.09)	2.94 (0.09)	4.87 (0.14)	3.50 (0.12)	2.88 (0.06)	1.93 (0.05)	3.68 (0.08)	2.50 (0.07)	4.36 (0.08)	2.86 (0.06)
STFGNN	3.33 (0.08)	2.33 (0.06)	4.19 (0.06)	2.96 (0.06)	4.90 (0.12)	3.47 (0.07)	2.89 (0.04)	1.97 (0.07)	3.67 (0.07)	2.50 (0.09)	4.27 (0.07)	2.89 (0.08)
DSTAGNN	3.31 (0.19)	2.30 (0.14)	4.15 (0.12)	2.99 (0.11)	4.90 (0.16)	3.47 (0.10)	3.00 (0.08)	1.99 (0.05)	3.65 (0.12)	2.48 (0.10)	4.32 (0.09)	2.91 (0.07)
FOGS	3.26 (0.09)	2.27 (0.09)	4.21 (0.12)	2.88 (0.11)	4.71 (0.11)	3.29 (0.09)	2.85 (0.08)	1.96 (0.07)	3.68 (0.10)	2.38 (0.08)	4.28 (0.12)	2.86 (0.10)
STEP	3.19 (0.05)	2.26 (0.04)	3.91 (0.08)	2.75 (0.07)	4.31 (0.11)	3.15 (0.10)	2.80 (0.05)	1.98 (0.04)	3.47 (0.05)	2.38 (0.06)	**3.77 (0.07)**	2.52 (0.06)
AdaRNN	3.24 (0.08)	2.26 (0.07)	4.05 (0.08)	2.95 (0.08)	4.90 (0.08)	3.50 (0.07)	2.86 (0.06)	1.98 (0.06)	3.57 (0.07)	2.44 (0.06)	4.20 (0.08)	2.84 (0.07)
ST-GFSL	3.37 (0.12)	2.29 (0.09)	4.20 (0.10)	2.86 (0.09)	4.68 (0.12)	3.28 (0.10)	2.97 (0.09)	2.00 (0.08)	3.73 (0.07)	2.40 (0.07)	4.25 (0.10)	2.76 (0.07)
TPB	**3.05 (0.04)****	**2.10 (0.04)****	**3.80 (0.06)****	**2.65 (0.04)****	**4.30 (0.07)****	**3.02 (0.06)****	**2.68 (0.04)****	**1.80 (0.03)****	**3.32 (0.05)****	**2.22 (0.07)****	3.80 (0.06)**	**2.50 (0.07)****

- **DCRNN** [24]: A spatial-temporal network that uses diffusion techniques and RNN.
- **GWN** [45]: Graph WaveNet utilizes the adaptive adjacency matrix and dilated causal convolution to capture the spatial temporal dependency.
- **STFGNN** [23]: A network that uses DTW distance to construct a temporal graph and fuse the spatial and temporal information in a gated module.
- **DSTAGNN** [20]: A network that constructs spatial-temporal aware graph and uses K-th order Chebyshev polynomial to aggregate the information.
- **FOGS** [35]: A framework that uses node2vec to learn spatial-temporal correlation and predicts the first-order gradients.

- Pre-training traffic forecasting method
  - **STEP** [37]: A pre-training framework that reconstructs the large-ratio masked traffic data based on transformer layers.
- Cross-city traffic forecasting methods
  - **AdaRNN** [8]: A time series transfer learning framework that reduces the distribution mismatch between time series to make model more adaptive.
  - **ST-GFSL** [29]: A state-of-art few-shot traffic forecasting framework that learns the metaknowledge of traffic nodes to generate the parameter of linear and convolution layers.

Not that baselines such as STrans-GAN [49], CrossTReS [18] could not be fairly compared here since they utilize multimodal data while our setting is cross-city univariate forecasting. The reason to do univariate forecasting is that we would like to tackle the situation that the target city has very few univariate data and data from other modalities is missing.

Metric: Two commonly used metrics in traffic forecasting are used to evaluate the performance of all baselines, including Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). The formulas are as follows.

$$RMSE = \sqrt{\frac{1}{s}\sum_{i=1}^{s}(y_i - \hat{y}_i)^2}, \quad MAE = \frac{1}{s}\sum_{i=1}^{s}|y_i - \hat{y}_i|$$

Results: The overall result is shown in Table 2. The following observations could be drawn: (1) Our proposed TPB achieves superior performance compared to the baselines not only in short-term but also in long-term forecasting. (2) Traditional methods have bad performance because the forecasting task has strong non-linearity, which is hard to be captured by these methods. (3) Compared with the typical deep-learning traffic forecasting methods implemented in the *Reptile* framework, TPB achieves a performance enhancement of 9.92% and 13.13% in RMSE and MAE respectively on average. The difference between these methods and our framework is that we add traffic-pattern-based metaknowledge in the forecasting stage. So, the significant performance enhancement indicates that the additional metaknowledge offered by the traffic pattern bank does help the model forecast more precisely. Further experiments about the effectiveness of our proposed traffic pattern bank would be shown in Sec. 5.3 (4) Compared with the pre-training traffic forecasting

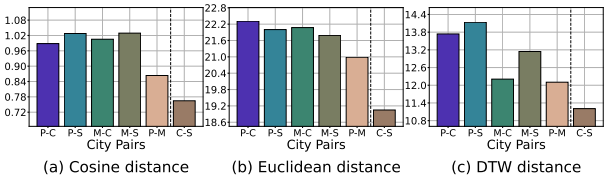

Figure 3: The distance between each pair of one-day raw traffic patches (time series of length 288) of four datasets. For simplicity, datasets are denoted by their first letter, e.g., "C-S" indicates *Chengdu* and *Shenzhen*.

method, TPB achieves a performance enhancement of 4.26% and 8.96% in RMSE and MAE respectively. This indicates that directly using the pre-trained transformer encoder is not enough and refining the metaknowledge in encoder space to construct the traffic pattern bank does help the model make accurate predictions in the few-shot traffic forecasting task. (5) Compared with the cross-city traffic forecasting methods, TPB achieves a performance enhancement of 6.52% and 7.71% in RMSE and MAE respectively on average and outperforms these methods on all datasets. The result shows that the traffic pattern bank of the TPB framework captures better metaknowledge than the unique design of ST-GFSL and AdaRNN.

Data Distribution Analysis: Fig. 3 shows the dataset distance between *Chengdu* and *Shenzhen* is smaller than other dataset pairs on all of the three metrics and Table 1 shows the mean and std between *Chengdu* and *Shenzhen* are more similar. So, The data distributions of *Chengdu* and *Shenzhen* are more similar than other pairs of cities. Consequently, the setting of P,M,S→Chengdu and P,M,C→Shenzhen represent the setting that knowledge transfers to target city with similar distribution while M,C,S→PEMS-BAY and P,C,S→METR-LA represent the setting that knowledge transfers to a target city with different distribution. We could observe that TPB outperforms other methods in both settings in Table 2. This demonstrates the superiority and robustness of TPB in the cross-city forecasting scenario.

5.3 RQ2: TPB Improves Different STmodel

TPB is a framework for few-shot traffic forecasting that different STmodels are able to plug into this framework. Specifically, TPB offers the reconstructed adjacency matrix A and metaknowledge embedding M based on traffic pattern bank to help the STmodel forecasting precisely. In order to verify the effectiveness of the traffic pattern bank on the few-shot setting, we implement several advanced STmodels into the TPB framework and check the performance enhancement. We have two types of training processes for these STmodels. (1) These STmodels are trained in *Reptile* meta-learning framework and the results are denoted as *STmodel-Reptile*. (2) These STmodels are plugged into the TPB framework and the training process of TPB is then conducted, the results of which are denoted as *STmodel-TPB*. By comparing the results of these two types of setting, we could know whether the traffic pattern bank help forecast future traffic data.

Fig. 4 shows the RMSE results of the aforementioned methods on four datasets of multi-step forecasting, from which we have the following observations: (1) Plugged into the TPB framework, all of

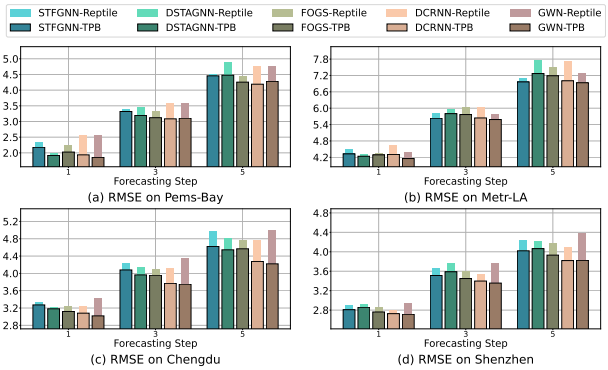

Figure 4: Performance comparison of different STmodels between being trained in *Reptile* meta-learning framework and being trained in TPB framework. RMSE of multi-step forecasting on four datasets is plotted.

the STmodels show improved performance compared to when they are trained in the *Reptile* meta-learning framework. This demonstrates the effectiveness of the traffic pattern bank in helping the STmodels to perform better on the few-shot traffic forecasting task. (2) The performance enhancement is more pronounced for longer forecasting horizons. This indicates the traffic pattern bank is able to capture more complex patterns that are important for longer-term forecasting. (3) Simple STmodels such as Graph Wavenet have shown more significant performance improvement than complex STmodels such as STFGNN and DSTAGNN. This indicates that simple STmodels are more flexible and the metaknowledge of traffic pattern bank adapts to these models more easily.

5.4 RQ3: How to choose K?

In the pattern generation phase, clustering parameter K significantly influences the quality of the generated traffic pattern bank. To quantitatively analyze the quality of the traffic pattern bank of different K, we calculate the Silhouette score [36] of the traffic pattern bank. It is calculated as follows.

$$s = \sum_{i=1}^{P} \frac{b_i - a_i}{\max(a_i, b_i)} \tag{15}$$

Here, P is the total number of traffic patches and a_i is the average distance between the traffic patch and all other traffic patches in its cluster and b_i is the average distance between the traffic patch and all other traffic patches in the nearest cluster. The Silhouette score ranges from -1 to 1, where a higher value indicates that the samples within each cluster are very similar to one another and thus the clusters are well-defined.

Fig. 5 shows two parts of the result. In (a)~(d), the Silhouette score of clustering results and RMSE of the 1-hop forecasting results of different K on four datasets are plotted. The following phenomena could be observed in these four plots. (1) When K is small (around 10), the Silhouette scores of the clustering result are large, and the RMSE of these results is small. (2) As K increases, the Silhouette score drops drastically, and the RMSE increases drastically at the same time. From these phenomena, we can say that the Silhouette

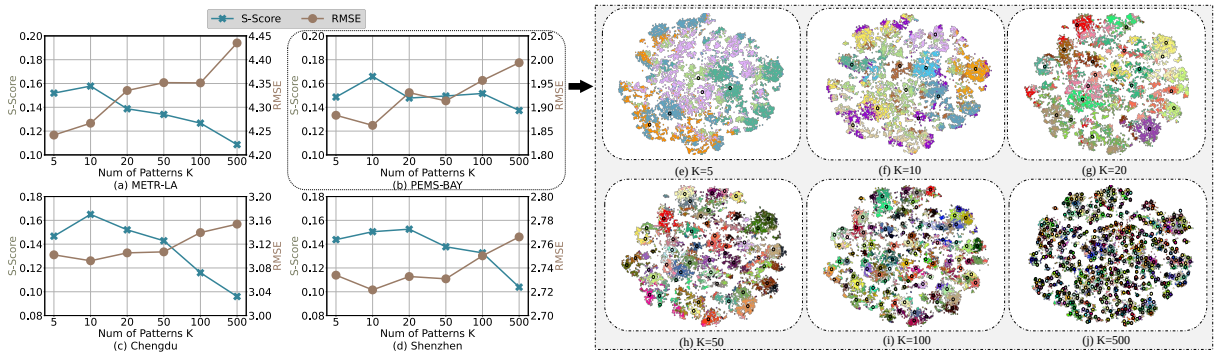

Figure 5: Patterns analysis. (a)~(d) The Silhouette score of clustering result and RMSE of the 1-hop forecasting results of different clustering parameters K on four datasets. A higher Silhouette score indicates the clusters are better defined and thus the traffic pattern bank is of better quality. (e)~(j) The TSNE visualization of the embeddings of traffic patch and pattern with different K on the PEMS-BAY dataset. The colored dots indicate the traffic patch embeddings belonging to different clusters. The black circles indicate the traffic pattern embeddings, which are the centroids of different clusters.

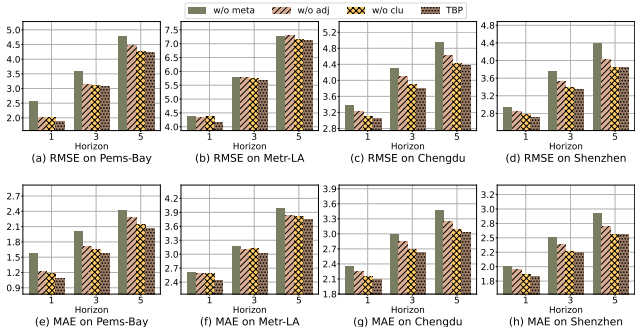

Figure 6: Ablation study of TPB. Both RMSE and MAE of multi-step forecasting of the variants are reported.

score is a good criterion for the quality of the traffic pattern bank given a K, and using a traffic pattern bank with a high Silhouette score in TBP could lead to a small error. In (e)~(i), the TSNE visualization of the embeddings of traffic patch and pattern with different K on the PEMS-BAY dataset is shown. From these plots, we could observe that the traffic patch embeddings belonging to different clusters are more distinct when a smaller value of K is used, while a larger value of K results in one raw cluster being divided into several traffic patterns. This indicates that a smaller value of K produces more well-defined clusters and a higher-quality traffic pattern bank. Furthermore, we could observe that the final result of TPB is inversely proportional to the Silhouette score. Consequently, it is efficient to select a proper K by calculating the Silhouette score and choosing the K with the highest Silhouette score.

5.5 RQ4: Ablation Study

In this section, we verify the effectiveness of each module of TPB. We remove or modify each module to get a variant of TPB and do grid search to get the best performance of these variants. There are three types of variants of TPB. (1) We remove traffic pattern bank and downstream metaknowledge. This variant is denoted as **w/o meta**. (2) We replace the metaknowledge-based adjacency matrix with the pre-defined static adjacency matrix, which is denoted as **w/o adj**. (3) We remove clustering process and randomly select K traffic patches as traffic pattern bank, which is denoted as **w/o clu**.

Fig. 6 shows the results of these three variants and compares them with the results of TPB. First, we can see that without the metaknowledge, the performance of the variant has significantly decreased, which demonstrates traffic pattern bank contains rich metaknowledge from the source cities and is helpful in few-shot traffic forecasting. Then, the variant that uses the predefined graph performs worse. This is not surprising because the predefined graph is not able to capture the dynamic and context-specific relationships between nodes, while the metaknowledge-based adjacency matrix is constructed based on the traffic pattern bank and is able to adapt to the specific spatial relation of the target city. Finally, the random-selected traffic pattern bank degrades the performance. This indicates the clustering process selects robust and representative traffic patterns from the source cities and the use of these patterns in the target city leads to improved performance.

6 CONCLUSION

In this work, we propose a novel cross-city few-shot traffic forecasting framework named TPB. We demonstrate that the traffic pattern is similar across cities. To capture the similarity, we pre-train a traffic patch encoder and use it to generate a traffic pattern bank from data-rich cities to help downstream cross-city traffic forecasting. Experiments on real-world traffic datasets demonstrate the superiority over state-of-the-art methods for cross-city few-shot traffic forecasting of the TPB framework. In the future, we will further investigate the usage of pattern bank on other few-shot settings.

ACKNOWLEDGEMENT

This work was sponsored by National Key Research and Development Program of China under Grant No.2022YFB3904204, National Natural Science Foundation of China under Grant No.62102246, No.62272301, No.62176243, and Provincial Key Research and Development Program of Zhejiang under Grant No.2021C01034.

REFERENCES

[1] Yasunori Akagi, Takuya Nishimura, Takeshi Kurashima, and Hiroyuki Toda. 2018. A Fast and Accurate Method for Estimating People Flow from Spatiotemporal Population Data.. In *IJCAI*. 3293–3300.

[2] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive graph convolutional recurrent network for traffic forecasting. *arXiv preprint arXiv:2007.02842* (2020).

[3] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. 2020. Spectral temporal graph neural network for multivariate time-series forecasting. *Advances in neural information processing systems* 33 (2020), 17766–17778.

[4] Jeongwhan Choi, Hwangyong Choi, Jeehyun Hwang, and Noseong Park. 2022. Graph neural controlled differential equations for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 6367–6374.

[5] Razvan-Gabriel Cirstea, Bin Yang, Chenjuan Guo, Tung Kieu, and Shirui Pan. 2022. Towards spatio-temporal aware traffic time series forecasting. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2900–2913.

[6] Zulong Diao, Xin Wang, Dafang Zhang, Yingru Liu, Kun Xie, and Shaoyao He. 2019. Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 890–897.

[7] GAIA initiative Didi. 2020. Didi Chuxing data. https://gaia.didichuxing.com.

[8] Yuntao Du, Jindong Wang, Wenjie Feng, Sinno Pan, Tao Qin, Renjun Xu, and Chongjun Wang. 2021. Adarnn: Adaptive learning and forecasting of time series. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 402–411.

[9] Shen Fang, Qi Zhang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2019. GSTNet: Global Spatial-Temporal Network for Traffic Flow Prediction.. In *IJCAI*. 2286–2293.

[10] Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. 2021. Spatial-temporal graph ode networks for traffic flow forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 364–373.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*. PMLR, 1126–1135.

[12] Kan Guo, Yongli Hu, Yanfeng Sun, Sean Qian, Junbin Gao, and Baocai Yin. 2021. Hierarchical Graph Convolution Network for Traffic Forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 151–159.

[13] Liangzhe Han, Bowen Du, Leilei Sun, Yanjie Fu, Yisheng Lv, and Hui Xiong. 2021. Dynamic and multi-faceted spatio-temporal deep learning for traffic speed forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 547–555.

[14] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16000–16009.

[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[16] Rongzhou Huang, Chuyin Huang, Yubao Liu, Genan Dai, and Weiyang Kong. 2020. LSGCN: Long Short-Term Traffic Prediction with Graph Convolutional Networks.. In *IJCAI*. 2355–2361.

[17] Jiahao Ji, Jingyuan Wang, Zhe Jiang, Jiawei Jiang, and Hu Zhang. 2022. STDEN: Towards Physics-guided Neural Networks for Traffic Flow Prediction. (2022).

[18] Yilun Jin, Kai Chen, and Qiang Yang. 2022. Selective Cross-City Transfer Learning for Traffic Prediction via Source City Region Re-Weighting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 731–741.

[19] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[20] Shiyong Lan, Yitong Ma, Weikang Huang, Wenwu Wang, Hongyu Yang, and Pyang Li. 2022. Dstagnn: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting. In *International Conference on Machine Learning*. PMLR, 11906–11917.

[21] Hyunwook Lee, Seungmin Jin, Hyeshin Chu, Hongkyu Lim, and Sungahn Ko. 2021. Learning to Remember Patterns: Pattern Matching Memory Networks for Traffic Forecasting. *arXiv preprint arXiv:2110.10380* (2021).

[22] Hung-yi Lee, Shang-Wen Li, and Ngoc Thang Vu. 2022. Meta Learning for Natural Language Processing: A Survey. *arXiv preprint arXiv:2205.01500* (2022).

[23] Mengzhang Li and Zhanxing Zhu. 2021. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 4189–4196.

[24] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).

[25] Chumeng Liang, Zherui Huang, Yicheng Liu, Zhanyu Liu, Guanjie Zheng, Hanyuan Shi, Yuhao Du, Fuliang Li, and Zhenhui Li. 2022. Cblab: Scalable traffic simulation with enriched data supporting. *arXiv preprint arXiv:2210.00896* (2022).

[26] Marco Lippi, Matteo Bertini, and Paolo Frasconi. 2013. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Trans. Intelligent Transportation Systems* 14, 2 (2013), 871–882.

[27] Xu Liu, Yuxuan Liang, Chao Huang, Hengchang Hu, Yushi Cao, Bryan Hooi, and Roger Zimmermann. 2023. Do We Really Need Graph Neural Networks for Traffic Forecasting? arXiv:2301.12603 [cs.LG]

[28] Zhanyu Liu, Chumeng Liang, Guanjie Zheng, and Hua Wei. 2023. FDTI: Fine-grained Deep Traffic Inference with Roadnet-enriched Graph. *arXiv preprint arXiv:2306.10945* (2023).

[29] Bin Lu, Xiaoying Gan, Weinan Zhang, Huaxiu Yao, Luoyi Fu, and Xinbing Wang. 2022. Spatio-Temporal Graph Few-Shot Learning with Cross-City Knowledge Transfer. *arXiv preprint arXiv:2205.13947* (2022).

[30] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* (2018).

[31] Ali Yadavar Nikravesh, Samuel A Ajila, Chung-Horng Lung, and Wayne Ding. 2016. Mobile network traffic prediction using MLP, MLPWD, and SVM. In *2016 IEEE International Congress on Big Data (BigData Congress)*. IEEE, 402–409.

[32] Iwao Okutani and Yorgos J Stephanedes. 1984. Dynamic prediction of traffic volume through Kalman filtering theory. *Transportation Research Part B: Methodological* 18, 1 (1984), 1–11.

[33] Boris N Oreshkin, Arezou Amini, Lucy Coyle, and Mark Coates. 2021. FC-GAGA: Fully connected gated graph architecture for spatio-temporal traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 9233–9241.

[34] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. 2019. Urban traffic prediction from spatio-temporal data using deep meta learning. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1720–1730.

[35] Xuan Rao, Hao Wang, Liang Zhang, Jing Li, Shuo Shang, and Peng Han. 2022. Fogs: First-order gradient supervision with learning-based graph for traffic flow forecasting. In *Proceedings of International Joint Conference on Artificial Intelligence, IJCAI*. ijcai. org.

[36] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.

[37] Zezhi Shao, Zhao Zhang, Fei Wang, and Yongjun Xu. 2022. Pre-training Enhanced Spatial-temporal Graph Neural Network for Multivariate Time Series Forecasting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1567–1577.

[38] Zezhi Shao, Zhao Zhang, Wei Wei, Fei Wang, Yongjun Xu, Xin Cao, and Christian S Jensen. 2022. Decoupled dynamic spatial-temporal graph neural network for traffic forecasting. *arXiv preprint arXiv:2206.09112* (2022).

[39] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems* 30 (2017).

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[41] Leye Wang, Xu Geng, Xiaojuan Ma, Feng Liu, and Qiang Yang. 2018. Cross-city transfer learning for deep spatio-temporal prediction. *arXiv preprint arXiv:1802.00386* (2018).

[42] Ying Wei, Yu Zheng, and Qiang Yang. 2016. Transfer knowledge between cities. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1905–1914.

[43] Billy M Williams and Lester A Hoel. 2003. Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *Journal of transportation engineering* 129, 6 (2003), 664–672.

[44] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. 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*. 753–763.

[45] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121* (2019).

[46] Huaxiu Yao, Yiding Liu, Ying Wei, Xianfeng Tang, and Zhenhui Li. 2019. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *The World Wide Web Conference*. 2181–2191.

[47] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. 2018. Revisiting Spatial-Temporal Similarity: A Deep Learning Framework for Traffic Prediction. arXiv:1803.01254 [cs.LG]

[48] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875* (2017).

[49] Yingxue Zhang, Yanhua Li, Xun Zhou, Xiangnan Kong, and Jun Luo. 2022. STrans-GAN: Spatially-Transferable Generative Adversarial Networks for Urban Traffic Estimation. In *2022 IEEE International Conference on Data Mining*. IEEE, 743–752.

[50] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 1234–1241.

[51] Zheng Zhu, Bo Peng, Chenfeng Xiong, and Lei Zhang. 2016. Short-term traffic flow prediction with linear conditional Gaussian Bayesian network. *Journal of Advanced Transportation* 50, 6 (2016), 1111–1123.