Urban Traffic Prediction from Spatio-Temporal Data Using Deep Meta Learning

Zheyi Pan[1], Yuxuan Liang[4], Weifeng Wang[1], Yong Yu[1], Yu Zheng[2,3,4], Junbo Zhang[2,3,5]*
[1]Department of Computer Science and Engineering, Shanghai Jiaotong University, China
[2]JD Intelligent Cities Research, China
[3]JD Intelligent Cities Business Unit, China
[4]School of Computer Science and Technology, Xidian University, China
[5]Institute of Artificial Intelligence, Southwest Jiaotong University, China
{zhpan,wfwang,yyu}@apex.sjtu.edu.cn;{yuxliang,msyuzheng,msjunbozhang}@outlook.com

ABSTRACT

Predicting urban traffic is of great importance to intelligent transportation systems and public safety, yet is very challenging because of two aspects: 1) complex spatio-temporal correlations of urban traffic, including spatial correlations between locations along with temporal correlations among timestamps; 2) diversity of such spatio-temporal correlations, which vary from location to location and depend on the surrounding geographical information, *e.g.*, points of interests and road networks. To tackle these challenges, we proposed a deep-meta-learning based model, entitled ST-MetaNet, to *collectively* predict traffic in all location at once. ST-MetaNet employs a sequence-to-sequence architecture, consisting of an encoder to learn historical information and a decoder to make predictions step by step. In specific, the encoder and decoder have the same network structure, consisting of a recurrent neural network to encode the traffic, a meta graph attention network to capture diverse spatial correlations, and a meta recurrent neural network to consider diverse temporal correlations. Extensive experiments were conducted based on two real-world datasets to illustrate the effectiveness of ST-MetaNet beyond several state-of-the-art methods.

CCS CONCEPTS

• **Information systems → Spatial-temporal systems**; **Data mining**; • **Computing methodologies → Neural networks**.

KEYWORDS

Urban traffic; Spatio-temporal data; Neural network; Meta learning

ACM Reference Format:
Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, Junbo Zhang. 2019. Urban Traffic Prediction from Spatio-Temporal Data Using Deep Meta Learning. In *The 25th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD'19), August 4–8, 2019, Anchorage, AK, USA.* ACM, NY, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330884

*Junbo Zhang is the corresponding author.

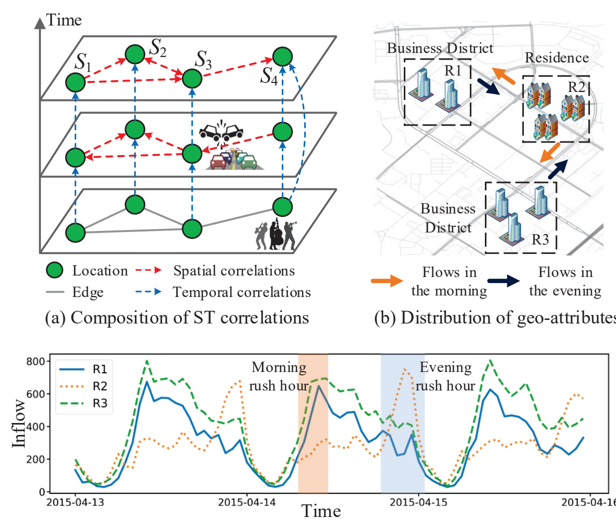
KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330884

1 INTRODUCTION

Recent advances in data acquisition technologies and mobile computing lead to a collection of large amounts of traffic data, such as vehicle trajectories, enabling much urban analysis and related applications [30]. Urban traffic prediction has become a mission-critical work for the development of a smart city, as it can provide insights for urban planning and traffic management to improve the efficiency of public transportation, as well as give early warnings for public safety emergency management [29]. However, predicting urban traffic is very challenging because of complicated spatio-temporal (ST) correlations.

Complex composition of ST correlations: Since traffic changes on the spatial domain (*e.g.* along road networks), we employ a geo-graph to describe the spatial structure, where each node denotes a location and each edge represents the relation between locations, as shown in Figure 1 (a). As urban traffic is highly dynamic, we consider such correlations from the following two factors.



(a) Composition of ST correlations (b) Distribution of geo-attributes

(c) Diversity of ST correlations

Figure 1: Challenges of urban traffic prediction.

• Spatial correlations. As the red arrow shown in Figure 1 (a), the traffic of some locations affects mutually and changes all the time. For example, traffic on road networks has strong spatial dependencies [14]. When a car accident occurs near S3, the traffic jam could quickly diffuse to its neighbors, i.e., S1, S2, and S4, on the geo-graph. Such impact brings challenges to forecasting the traffic.

• Temporal correlations. Given a location, the current readings of its traffic, such as inflows and outflows, are correlated with its precedents. For instance, the taxi flow of a region is affected by the recent values, both near and far [27]. When a special event occurs at S4 (*e.g.*, a large concert), many people would go into the location and such location status would hold for a long time.

As both types of correlations affect urban traffic, it is necessary to *simultaneously* model such spatial and temporal correlations.

Diversity of ST correlations: In a city, characteristics of locations and their mutual relationship are diverse, depending on their own geo-graph attributes: 1) *node attributes*: the surrounding environment of a location, *namely*, nearby points of interests (POIs) and density of road networks (RNs); 2) *edge attributes*: the relationship between two nodes, such as the connectivity of roads and the distance between them. As shown in Figure 1 (b), District R1, R2 and R3 have different distributions of POIs and road network structures. Consequently, the characteristics of these districts are different, and the trends of their inflows are diverse, as shown in Figure 1 (c), which demonstrates that districts with different characteristics always have different types of ST correlations.

Nonetheless, locations with a similar combination of geo-graph attributes can lead to similar characteristics and analogous types of ST correlations. In Figure 1 (b), District R1 and R3 contain numerous office buildings that indicate business districts, while R2 contains many apartments, denoting a residential district. In general, citizens usually commute from home to their workplaces in the morning, while return at night. Thus, business district R1 and R3 witness similar upward trends of inflows in the morning, while the residential district R2 meets a different rush hour in the evening, as shown in Figure 1 (c). Therefore, modeling inherent relationship between geo-graph attributes and ST correlations' types plays an essential role in urban traffic prediction.

Recently, although there has been significant growth of works for urban traffic prediction as well as analogous ST prediction tasks, the aforementioned challenges are still not tackled well. First of all, some works [15, 22] focus on modeling ST correlations by a single model for all locations. These methods cannot explicitly model the inherent relationships between geo-graph attributes and various types of ST correlations, as a result, such relationship is hard to be learned without any prior knowledge. Another group of works [16, 23, 28] adopt multi-task learning approaches, which mainly build multiple sub-models for each location and all of these sub-models are trained together by using similarity constraints between locations. These methods depend on the prior knowledge or strong assumption of the specific tasks, *i.e.*, the location similarity. However, such side information can only provide relatively weak supervision, producing unstable & tricked, even ruinous results in the complex real-world applications.

To tackle the aforementioned challenges, we propose a deep meta learning based framework, entitled ST-MetaNet, for urban traffic prediction. The main intuition is depicted in Figure 2. The urban traffic prediction is a typical ST forecasting problem, whose main factors include spatial and temporal correlations that are implicitly affected by the characteristics of nodes (locations) and edges (the mutual relation between two nodes). Intuitively, the characteristic of a node is influenced by its attributes, like GPS locations and nearby POIs. Likewise, the characteristic of an edge depends on its attributes, *e.g.*, road connectivity and the distance between locations. Based on these insights, ST-MetaNet first extracts the *meta knowledge* (*i.e.* characteristics) of nodes and edges from their attributes, respectively. The meta knowledge is then used to model the ST correlations, namely, generating weights of the prediction networks. The contributions of our work are four folds:

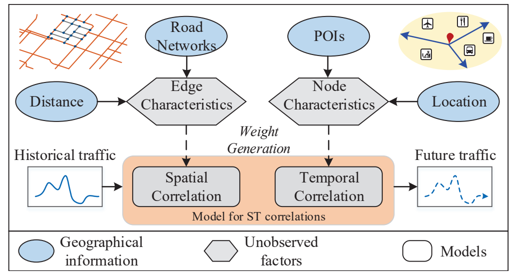


Figure 2: Insight of the framework.

• We propose a novel deep meta learning based model, named ST-MetaNet, to predict urban traffic. ST-MetaNet leverages the meta knowledge extracted from geo-graph attributes to generate the parameter weights of a graph attention network and a recurrent network within sequence-to-sequence architecture. As a result, it can capture the inherent relation between geo-graph attributes and diverse types of ST correlations.

• We propose a meta graph attention network to model spatial correlations. The attention mechanism can capture the dynamic mutual relationship between locations, with regard to their current states. In addition, the weights of attention network are generated by meta knowledge of nodes and edges extracted from geo-graph attributes, such that it can model the diverse spatial correlations.

• We propose a meta gated recurrent neural network, which generates all weights of a normal gated recurrent unit from the meta knowledge of each node. Thus each location has an unique model for its unique type of temporal correlations.

• We evaluate ST-MetaNet on two tasks: taxi flow prediction and traffic speed prediction. The experiment results verify that ST-MetaNet can significantly improve urban traffic prediction, and learn better traffic-related knowledge from geo-graph attributes.

2 PRELIMINARIES

In this section, we introduce the definitions and the problem statement. For brevity, we present a table of notations in Table 1.

Table 1: Notations.

Notations	Description
N_l, N_t	Number of locations/timestamps.
τ_{in}, τ_{out}	The number of timestamps for historical/future traffic
$\mathcal{X} = \{X_t\}$	The traffic readings at all timestamps.
$\mathcal{V} = \{v^{(i)}\}$	The node attributes of location i.
$\mathcal{E} = \{e^{(ij)}\}$	The edge attributes between location i and j.
$\mathcal{N}_i$	Neighborhoods of location i.
$\text{NMK}(\cdot)$	The function to learn meta knowledge of node.
$\text{EMK}(\cdot)$	The function to learn meta knowledge of edge.
$g_\theta(\cdot)$	The function to generate weights of parameter θ.

Suppose there are N_l locations, which report D_t types of traffic information on N_t timestamps respectively.

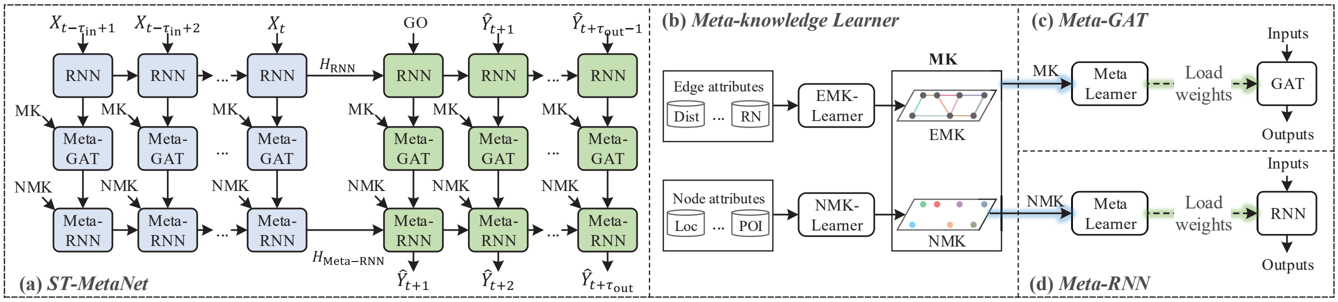


Figure 3: Overview of ST-MetaNet.

Definition 1. ***Urban traffic***. *The urban traffic is denoted as a tensor* $\mathcal{X} = (X_1, ..., X_{N_t}) \in \mathbb{R}^{N_t \times N_l \times D_t}$, *where* $X_t = (x_t^{(1)}, ..., x_t^{(N_l)})$ *denotes all locations' traffic information at timestamp* t.

Definition 2. ***Geo-graph attribute***. *Geo-graph attributes represent locations' surrounding environment and their mutual relations, which corresponds to node attributes and edge attributes respectively. Formally, let* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ *represent a directed graph, where* $\mathcal{V} = \{v^{(1)}, ..., v^{(N_l)}\}$ *and* $\mathcal{E} = \{e^{(ij)} | 1 \leq i, j \leq N_l\}$ *are lists of vectors denoting the geographical features of locations and relations between locations, respectively. Moreover, we use* $\mathcal{N}_i = \{j | e^{(ij)} \in \mathcal{E}\}$ *to denotes the neighbors of node* i.

Problem 1. *Given previous* τ_{in} *traffic information* $(X_{t-\tau_{in}+1}, ..., X_t)$ *and the geo-graph attributes* $\mathcal{G}$, *predict the traffic information for all locations in the next* τ_{out} *timestamps, denoted as* $(\hat{Y}_{t+1}, ..., \hat{Y}_{t+\tau_{out}})$.

3 METHODOLOGIES

In this section, we describe the architecture of ST-MetaNet for traffic prediction, as shown in Figure 3 (a). Following the sequence-to-sequence (Seq2Seq) architecture [18], ST-MetaNet is composed of two separate modules: the encoder (blue parts) and the decoder (green parts). The former one is used to encode the sequence of input, *i.e.*, historical information of urban traffic $\{X_{t-\tau_{\text{in}}+1}, \cdots, X_t\}$, producing the hidden states $\{H_{\text{RNN}}, H_{\text{Meta-RNN}}\}$, which are used as the initial states of the decoder that further predicts the output sequence $\{\hat{Y}_{t+1}, ..., \hat{Y}_{t+\tau_{\text{out}}}\}$.

More specifically, the encoder and the decoder have the same network structures, consisting of four components:

1) ***RNN (recurrent neural network)***. We employ RNN to embed the sequence of historical urban traffic, capable of learning long range temporal dependencies.

2) ***Meta-knowledge learner***. As shown in Figure 3 (b), we use two fully connected networks (FCNs), named node-meta-knowledge learner (NMK-Learner) and edge-meta-knowledge learner (EMK-Learner), to respectively learn the *meta-knowledge* of nodes (NMK) and edges (EMK) from node attributes (*e.g.*, POIs and GPS locations) and edge attributes (*e.g.*, road connectivity and the distance between locations). Then the learned meta knowledge is further used to learn the weights of another two networks, *i.e.*, graph attention network (GAT) and RNN. Taking a certain node as an example, the attributes of the node are fed into the NMK-Learner, and it outputs a vector, representing the meta knowledge of that node.

3) ***Meta-GAT (meta graph attention network)*** is comprised of a meta learner and a GAT, as shown in Figure 3 (c). we propose employing a FCN as the meta learner, whose inputs are meta knowledge of all nodes and edges, and outputs are the weights of GAT. Meta-GAT can capture diverse spatial correlations by individually broadcasting locations' hidden states along edges.

4) ***Meta-RNN (meta recurrent neural network)*** is comprised of a meta learner and a RNN, as shown in Figure 3 (d). The meta learner here is a typical FCN whose inputs are meta knowledge of all nodes and outputs are the weights of RNN for each location. Meta-RNN can capture diverse temporal correlations associated with the geographical information of locations.

3.1 Recurrent Neural Network

To encode the temporal dynamics of urban traffic, we employ a RNN layer as the first layer of Seq2Seq architecture. There are various types of RNN implementation for time series analysis. Among them, as gated recurrent unit (GRU) [9] is a simple but effective structure, we introduce GRU as a concrete example to illustrate ST-MetaNet. Formally, a GRU is defined as:

$$h_t = \text{GRU}(z_t, h_{t-1} | W_\Omega, U_\Omega, b_\Omega), \tag{1}$$

where $z_t \in \mathbb{R}^D$ and $h_t \in \mathbb{R}^{D'}$ are the input vector and the encoding state at timestamp t, respectively. $W_\Omega \in \mathbb{R}^{D' \times D}$ and $U_\Omega \in \mathbb{R}^{D' \times D'}$ are weight matrices. $b_\Omega \in \mathbb{R}$ are biases ($\Omega \in \{u, r, h\}$). GRU derives the vector representations of a hidden state, which is expressed as:

$$\begin{aligned} u &= \sigma(W_u z_t + U_u h_{t-1} + b_u) \\ r &= \sigma(W_r z_t + U_r h_{t-1} + b_r) \\ h_t &= u \circ h_{t-1} + (1-u) \circ \phi(W_h z_t + U_h (r \circ h_{t-1}) + b_h), \end{aligned} \tag{2}$$

where $\circ$ is the element-wise multiplication, $\sigma(\cdot)$ is sigmoid function, and $\phi(\cdot)$ is tanh function.

In urban traffic prediction, we collectively encode the traffic of all nodes (locations). To make nodes' encoding states in the same latent embedding space such that we can quantify the relations between them, RNN networks for all nodes share the same parameters. Suppose the input is $Z_t = (z_t^{(1)}, ..., z_t^{(N_l)})$ where $z_t^{(i)}$ is the input of node i at timestamp t, we obtain hidden states for all nodes, denoted as $H_t = (h_t^{(1)}, ..., h_t^{(N_l)})$, by the following formula:

$$h_t^{(i)} = \text{GRU}(z_t^{(i)}, h_{t-1}^{(i)} | W_\Omega, U_\Omega, b_\Omega), \quad \forall i \in \{1, ..., N_l\}, \tag{3}$$

More specifically, in ST-MetaNet the inputs of RNNs in the encoder and decoder are $X_t = (x_t^{(1)}, ..., x_t^{(N_l)})$ and $\hat{Y}_t = (\hat{y}_t^{(1)}, ..., \hat{y}_t^{(N_l)})$, respectively, as shown in Figure 3 (a).

3.2 Meta-Knowledge Learner

In urban area, characteristics of locations and their mutual relationship are diverse, depending on geographical information, *e.g.*, POIs

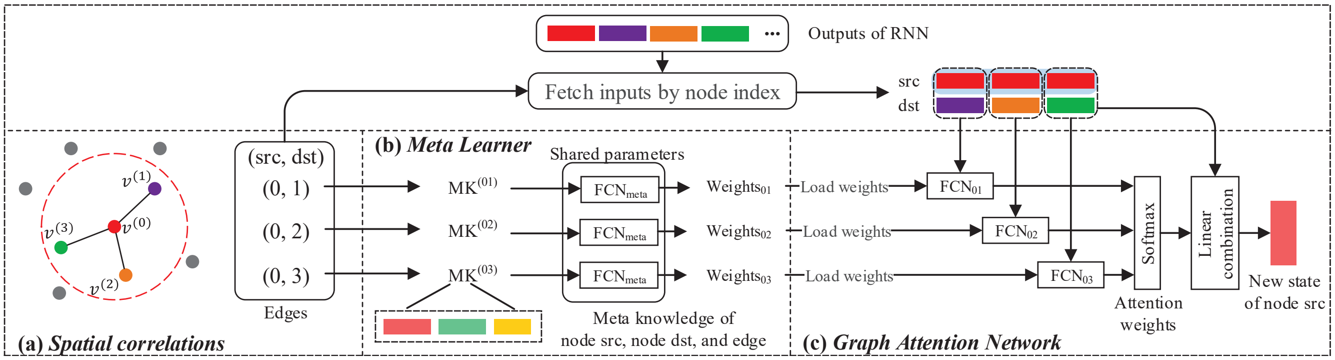


Figure 4: Structure of meta graph attention network.

and RNs. Such diverse characteristics bring about various types of ST correlations within urban traffic. Hence, we propose two meta-knowledge learners to learn traffic-related embeddings (meta knowledge) from geographical information, *i.e.*, NMK-Learner and EMK-Learner. As shown in Figure 3 (b), two meta-knowledge learners respectively employ different FCNs, in which input is the attribute of a node or an edge, and the corresponding output is the embedding (vector representation) of that node or edge. Since such embeddings are used for generating weights of GAT and RNN to capture ST correlations of urban traffic, the learned embeddings can reflect traffic-related characteristics of nodes and edges. For simplicity, we use $\mathrm{NMK}(v^{(i)})$ and $\mathrm{EMK}(e^{(ij)})$ to denote the learned meta knowledge (embedding) of a node and an edge, respectively.

3.3 Meta Graph Attention

Urban traffic has spatial correlations that some locations are mutually affected and such impact is highly dynamic. In addition, spatial correlations between two nodes are related to their geographical information, and such correlations are diverse from node to node and edge to edge. Inspired by graph attention network (GAT) [19], we propose employing attention mechanism into the framework to capture dynamic spatial correlations between nodes. However, it is inappropriate to directly apply GAT because all nodes and edges would use the same attention mechanism, ignoring the relationship between geographical information and spatial correlations.

To capture diverse spatial correlations, we propose a meta graph attention network (Meta-GAT) as shown in Figure 4, which employs attention network whose weights are generated from the meta knowledge (embeddings) by the meta learner. Consequently, the networks for spatial correlation modeling are different from node to node and edge to edge.

Formally, suppose the inputs of meta graph attention network are $H = (h^{(1)}, ..., h^{(N_l)}) \in \mathbb{R}^{N_l \times D_h}$ (*i.e.*, outputs of RNN at each timestamp) and geo-graph attributes $G = \{\mathcal{V}, \mathcal{E}\}$, while the output is $\bar{H} = (\bar{h}^{(1)}, ..., \bar{h}^{(N_l)}) \in \mathbb{R}^{N_l \times D_h}$, where D_h is the dimension of nodes' hidden states. The meta graph attention mechanism for each node mainly contains 2 steps: 1) attention score calculating for each edge; and 2) hidden state aggregation. As shown in Figure 4, we give an example to show Meta-GAT, that calculates the impact to the red node from its neighborhoods (the purple, orange, and green node) along edges. The details of Meta-GAT are as follows:

• Attention score calculation. The score of edge ij is related to the hidden states of node i and node j, as well as the meta knowledge of nodes and edge learned from geographical information. As shown in Figure 4, for edge ij, the first step is fetching the hidden states of nodes by index, i.e., $h^{(i)}$ and $h^{(j)}$, and meta knowledge $\mathrm{MK}^{(ij)}$, which is a composition of meta knowledge of nodes and edge:

$$\mathrm{MK}^{(ij)} = \mathrm{NMK}(v^{(i)}) \parallel \mathrm{NMK}(v^{(j)}) \parallel \mathrm{EMK}(e^{(ij)}),$$

where $\parallel$ is the concatenation operator. Then we apply a function to calculate the attention score, denoted as:

$$w^{(ij)} = a(h^{(i)}, h^{(j)}, \mathrm{MK}^{(ij)}) \in \mathbb{R}^{D_h},$$

where $w^{(ij)}$ is a D_h dimension vector, denoting the importance of how $h^{(j)}$ impacts $h^{(i)}$ at each channel. Like GAT [19] shown in Figure 4 (c), we employ a single fully connected layer to calculate function $a(\cdot)$. However, different pairs of nodes have different node and edge attributes, resulting in different patterns of edge attention given the nodes' states. To model such diversity, we employ an edge-wise fully connected layer, followed by an activation of LeakyRelu:

$$a(h^{(i)}, h^{(j)}, \mathrm{MK}^{(ij)}) = \mathrm{LeakyReLU}(W^{(ij)}[h^{(i)} \parallel h^{(j)}] + b^{(ij)}),$$

where $W^{(ij)} \in \mathbb{R}^{D_h \times 2D_h}$, $b^{(ij)} \in \mathbb{R}$ are edge-wise parameters of the fully connected layer. In particular, these parameters are generated by the meta learner from the meta knowledge as shown in Figure 4 (b). Formally, let g_W and g_b be FCNs within the meta learner to generate $W^{(ij)}$ and $b^{(ij)}$ respectively, then for any edge (i, j):

$$W^{(ij)} = g_W(\mathrm{MK}^{(ij)}) \in \mathbb{R}^{2D_h \times D_h}$$
$$b^{(ij)} = g_b(\mathrm{MK}^{(ij)}) \in \mathbb{R}.$$

Note that the output of a FCN is a vector, so we need to reshape the output to the corresponding parameter shape.

• Hidden state aggregation. Like GAT, we firstly normalize the attention score for a node across all its neighborhoods by softmax:

$$\alpha^{(ij)} = \frac{\exp(w^{(ij)})}{\sum_{j \in \mathcal{N}_i} \exp(w^{(ij)})}.$$

Then for each node, we calculate the overall impact of neighborhoods by linear combinations of the hidden states corresponding to the normalized weights and apply a nonlinearity function σ (*e.g.*, ReLU), which is expressed as $\sigma(\sum_{j \in \mathcal{N}_i} \alpha^{(ij)} h^{(j)})$. After that, let $\lambda^{(i)} \in (0, 1)$ be a trainable parameter denoting the weights of neighborhoods' impact to node i. And finally, the hidden state for node i with consideration of spatial correlations is expressed as:

$$\bar{h}^{(i)} = (1 - \lambda^{(i)}) h^{(i)} + \lambda^{(i)} \sigma(\sum_{j \in \mathcal{N}_i} \alpha^{(ij)} h^{(j)})$$

Since we extract meta knowledge from features of locations and edges, and use such information to generate the weights of graph attention network, Meta-GAT can model the inherent relationship between geo-graph attributes and diverse spatial correlations.

3.4 Meta Recurrent Neural Network

Since temporal correlations of urban traffic vary from location to location, a simple shared RNN (like Eq. 3) is not sufficient to capture diverse temporal correlations. To model such diversity, we adopt the similar idea of Meta-GAT to generate the weights of RNN from node embeddings, which is learned by NMK-Learner from node attributes (*e.g.,* POIs and RNs).

Here we introduce Meta-GRU as a concrete example of Meta-RNN. It adopts the node-wise parameters within GRU. Formally, we define Meta-GRU as:

$$H_t = \text{Meta-GRU}(Z_t, H_{t-1}, \mathcal{V}),$$

where $Z_t = (z_t^{(1)}, ..., z_t^{(N_l)})$ and $H_t = (h_t^{(1)}, ..., h_t^{(N_l)})$ are the inputs and the hidden states at timestamp t, respectively, and $\mathcal{V} = \{v^{(1)}, ..., v^{(N_l)}\}$ is the node attributes. Then for any node i, the current hidden states can be calculated by:

$$\begin{aligned}
W_\Omega^{(i)} &= g_{W_\Omega}(\text{NMK}(v^{(i)})) \\
U_\Omega^{(i)} &= g_{U_\Omega}(\text{NMK}(v^{(i)})) \\
b_\Omega^{(i)} &= g_{b_\Omega}(\text{NMK}(v^{(i)})) \\
h_t^{(i)} &= \text{GRU}(z_t^{(i)}, h_{t-1}^{(i)} | W_\Omega^{(i)}, U_\Omega^{(i)}, b_\Omega^{(i)}),
\end{aligned}$$

where $W_\Omega^{(i)}$,$U_\Omega^{(i)}$ and $b_\Omega^{(i)}$ are node-wise parameters generated from $v^{(i)}$ by a meta learner which consists of three types of FCNs g_{W_Ω}, g_{U_Ω}, g_{b_Ω} ($\Omega \in \{u, r, h\}$). As a result, all nodes have their individual RNNs respectively, and the models represent the temporal correlations related to the node attributes. In ST-MetaNet, we take the outputs of Meta-GAT as the inputs of Meta-RNN (*i.e.,* Z_t), accordingly, both diverse spatial and temporal correlations are captured.

3.5 Algorithm of Optimization

Suppose the loss function for framework training is $\mathcal{L}_{\text{train}}$, which measures the difference between the prediction values and the ground truth. We can train ST-MetaNet end-to-end by backpropagation. In specific, there are two types of trainable parameters. Let ω_1 denote the trainable parameters in common neural networks (Section 3.1), ω_2 denote all trainable parameters in the meta-knowledge learners (Section 3.2), and meta learners (Section 3.3 and Section 3.4). For parameter ω_1 in common neural networks, the gradient of it is $\nabla_{\omega_1}\mathcal{L}_{\text{train}}$. As for parameter ω_2 in meta-knowledge learner and meta learner which generates parameter θ, the gradient of ω_2 can be calculated by chain rule because both meta-knowledge learner and meta learner are differentiable neural networks:

$$\nabla_{\omega_2}\mathcal{L}_{\text{train}} = \nabla_\theta \mathcal{L}_{\text{train}} \nabla_{\omega_2}\theta.$$

Algorithm 1 outlines the training process of ST-MetaNet. We first construct training data (Lines 1-5). Then we iteratively optimizer ST-MetaNet by gradient descent (Lines 7-12) until the stopping criteria is met.

Algorithm 1: Training algorithm of ST-MetaNet

input : Urban traffic $\mathcal{X} = (X_1, \ldots, X_{N_t})$.
Node attributes $\mathcal{V} = \{v^{(1)}, \ldots, v^{(N_l)}\}$.
Edge attributes $\mathcal{E} = \{e^{(ij)}\}$.

1 $\mathcal{D}_{\text{train}} \leftarrow \emptyset$
2 **for** *available* $t \in \{1, \ldots, N_t\}$ **do**
3 $\quad X_{\text{input}} \leftarrow (X_{t-\tau_{\text{in}}+1}, \ldots, X_t)$
4 $\quad Y_{\text{label}} \leftarrow (X_{t+1}, \ldots, X_{t+\tau_{\text{out}}})$
5 $\quad$ put $\{X_{\text{input}}, \mathcal{V}, \mathcal{E}, Y_{\text{label}}\}$ into $\mathcal{D}_{\text{train}}$
6 initialize all trainable parameters
7 **do**
8 $\quad$ randomly select a batch $\mathcal{D}_{\text{batch}}$ from $\mathcal{D}_{\text{train}}$
9 $\quad$ forward-backward on $\mathcal{L}_{\text{train}}$ by $\mathcal{D}_{\text{batch}}$
10 $\quad \omega_1 = \omega_1 - \alpha \nabla_{\omega_1} \mathcal{L}_{\text{train}}$ // α is learning rate
11 $\quad \omega_2 = \omega_2 - \alpha \nabla_{\omega_2} \mathcal{L}_{\text{train}}$
12 **until** *stopping criteria is met*
13 output learned ST-MetaNet model

4 EVALUATION

4.1 Experimental Settings

Task Descriptions. We conduct experiments on two real-world tasks of urban traffic prediction. We first introduce the two tasks, and then illustrate the details of the datasets shown in Table 2.

Table 2: Details of the datasets.

Tasks	Taxi flow prediction	Speed prediction
Prediction target	inflow & outflow	traffic speed
Timespan	2/1/2015 - 6/2/2015	3/1/2012 - 6/30/2012
Time interval	1 hour	5 minutes
# timestamps	3600	34272
# nodes	1024	207
# edges	4114	3726
# node features	989	20
# edge features	32	1

1) ***Taxi flow prediction***. We adopt grid-based flow prediction task [27] to evaluate our framework, where grids are regarded as nodes. We partition the Beijing city into 32×32 grids, and then obtain hourly taxi flows of each grid and the geo-graph attributes. The details of the datasets are as follow:

- *Taxi flow*. We obtain taxi flows from TDrive dataset [25], which contains a large amount of taxicab trajectories from Feb. 1st 2015 to Jun. 2nd 2015. For each grid, we calculate the hourly inflows and outflows from these trajectories by counting the number of taxis entering or exiting the grid.
- *Geo-graph attributes*. The node attributes of each grid consist of many features of POIs and RNs within it, including the number of POIs in each category, the number of roads and lanes, etc. The edge attributes are the features of roads connecting pairs of grids, e.g., the number of roads and lanes connecting two grids, etc.

In this task, we use previous 12-hour flows to predict the next 3-hour flows. We partition the data on the time axis into non-overlapping training, evaluation, and test data, by the ratio of 8:1:1.

2) ***Traffic Speed prediction***. In this task, we predict the traffic speed on road networks. The details of the datasets are as follow:

• *Traffic speed.* We adopt METR-LA dataset [12], which is collected from loop detectors in the highway of Los Angeles County, and then further processed and public by [14]. The data contains traffic speed readings of 207 sensors (nodes). The readings are aggregated into 5-minute windows.
• *Geo-graph attributes.* Since we do not have POI information, we make use of locations and road networks as features of geo-graph. The node attributes consist of GPS points of nodes, and road structure information for each node, i.e., a vector reflecting the road distance between the node and its k-nearest neighbors. The edge attribute is simply defined as the road distance between nodes. For the efficiency of model training and testing, we only keep the edge between each node and its k-nearest neighbors. Since the traffic correlations are directional on road networks [14], we collect node attributes and edge attributes on both directions respectively.

In this task, we set $k = 8$, and use previous 60-minute traffic speed to predict speed in the next 60 minutes. We partition the traffic speed dataset on the time axis into non-overlapping training, evaluation, and test data, by the ratio of 7:1:2.

Metrics. We use mean absolute error (MAE) and rooted mean square error (RMSE) to evaluate ST-MetaNet:

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

where n is the number of instances, $\hat{y}_i$ is the prediction result and y_i is the ground truth.

Baselines. We compare ST-MetaNet with the following baselines:
• **HA**. Historical average. We model the urban traffic as the seasonal process, where the period is one day. We use the average values of the previous seasons as the prediction result.
• **ARIMA**. Autoregressive integrated moving average is a widely used model for time series prediction, which combines moving average and autoregression. In the experiments, we train an individual ARIMA model for each node, and predict future steps separately.
• **GBRT**. Gradient Boosting Regression Tree is a non-parametric statistical learning method for regression problem. For each future step (e.g., next 1 hour or next 2 hour), we train a single GBRT, and predict the urban traffic, where the input consists of previous traffic information and node attributes.
• **Seq2Seq** [18]. We implement a two-layer sequence-to-sequence model for urban traffic prediction, where GRU is chosen as the RNN implementation. The features of nodes, i.e., node attributes, are added as the additional inputs. All nodes share the same model.
• **GAT-Seq2Seq** [19]. We employ graph attention network and Seq2Seq to model spatial and temporal correlations, respectively. It applies the similar structure as ST-MetaNet, which consists of a GRU layer, a GAT layer, and a GRU layer. The node attributes, are firstly embedded by FCN and then added as the additional inputs.
• **ST-ResNet** [27]. The model is widely used in grid-based flow prediction task. It models the spatio-temporal correlations by residual unit . We further fuse its output with node attributes, i.e., features of grids, and then make predictions.
• **DCRNN** [14]. This model is state-of-the-art in traffic prediction on road networks. it employs the diffusion convolution operator and Seq2Seq to capture spatial and temporal correlations, respectively.

Framework Settings. The structure settings of ST-MetaNet contains the following three parts:
• Structures of NMK-Learner and EMK-Learner. We simply employ two FCNs (2 layers with the same number of hidden units) as NMK-Learner and EMK-Learner respectively, to learn the meta knowledge of nodes and edges. We conduct grid search on the number of hidden units over $\{4, 8, 16, 32, 64\}$.
• Dimension of hidden states in Seq2Seq architecture. For simplicity, we use the same number of hidden units in all components (GRU, Meta-GAT, Meta-GRU) within the encoder and decoder, and conduct grid search on the number over $\{16, 32, 64, 128\}$.
• Weight generation of meta learners. For each generated parameters in Meta-GAT and Meta-GRU, i.e., $\mathbf{W}^{(ij)}$, $b^{(ij)}$, $\mathbf{W}_\Omega$, $\mathbf{U}_\Omega$, and b_Ω, we simply build a FCN with hidden units $[16, d_g, n]$ to generate parameter weights from the meta knowledge, where n is the number of parameters in the target. We search on d_g over $\{1, 2, 4, 8\}$.

The framework is trained by Adam optimizer [13] with gradient clipping. To tackle the discrepancy between training and inference in Seq2Seq architecture, we employ inverse sigmoid decay for schedule sampling [2] in the training phase.

4.2 Performance Results

The performance of the competing baselines and ST-MetaNet are shown in Table 3. In addition, we also list the number of trainable parameters involved in deep models to show the model complexity. For deep models, we train and test each of them five times, and present results as the format: "mean ±standard deviation".

In the task of taxi flow prediction, state-of-the-art (SOTA) refers to ST-ResNet. We list the overall performance and the performance of prediction in each hour, respectively. As shown in Table 3, ST-MetaNet outperforms all the baselines on both metrics. Specifically, ST-MetaNet shows over 9.6% and 5.8% improvements on overall MAE and RMSE beyond ST-ResNet (SOTA), respectively. The reasons are that ST-ResNet cannot utilize the road connection between grids, and all grids use the same convolutional kernel without considering the diversity of grid characteristics. Compared with GAT-Seq2Seq, ST-MetaNet also achieves significant improvement, because it explicitly models the inherent relationship between geographical information and ST correlations. Seq2Seq does not consider spatial correlations and diversity of temporal correlations, so it gets much larger error compared with other deep models. Conventional methods, i.e., ARIMA and GBRT, are not good enough to model urban traffic, due to the limitation of model expressiveness and the incapability to fully leverage geographical information.

In the task of traffic speed prediction, SOTA refers to DCRNN. We list the overall performance and the prediction results in the next 15 minute, 30 minute, and 60 minute (corresponding to future step 3,6, and 12, respectively). Similar to taxi flow prediction, ST-MetaNet significantly outperforms the conventional models, and simple deep models, i.e., Seq2Seq. GAT-Seq2Seq and DCRNN are two powerful models to simultaneously capture spatial and temporal correlations. However, ST-MetaNet considers the inherent relationship between geographical information and ST correlations in advance. Thus it can obtain lower errors than GAT-Seq2Seq and DCRNN.

In Meta-GAT and Meta-GRU, we generate the parameter weights defined in GAT and GRU, which would intuitively introduce more

Table 3: Performance results on taxi flow prediction and traffic speed prediction.

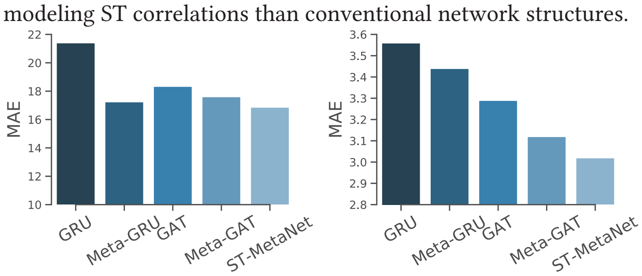
Models			HA	ARIMA	GBRT	Seq2Seq	GAT-SeqSeq	SOTA	**ST-MetaNet**
Taxi flow	overall	MAE	26.2	40.0	28.8	21.3 ± 0.06	18.3 ± 0.13	18.7 ± 0.53	**16.9** ± 0.13
		RMSE	56.5	86.8	60.9	42.6 ± 0.14	35.6 ± 0.23	36.1 ± 0.59	**34.0** ± 0.25
	1h	MAE	26.2	27.1	22.3	17.8 ± 0.05	16.3 ± 0.12	16.8 ± 0.50	**15.0** ± 0.14
		RMSE	56.5	58.3	47.7	35.1 ± 0.07	31.9 ± 0.21	31.9 ± 0.69	**29.9** ± 0.08
	2h	MAE	26.2	41.2	29.8	22.0 ± 0.06	18.7 ± 0.12	18.9 ± 0.57	**17.3** ± 0.14
		RMSE	56.5	77.0	62.6	43.6 ± 0.16	36.3 ± 0.20	36.4 ± 0.71	**34.7** ± 0.25
	3h	MAE	26.2	51.8	34.2	24.2 ± 0.09	19.9 ± 0.14	20.3 ± 0.52	**18.4** ± 0.10
		RMSE	56.5	108.0	70.3	48.1 ± 0.20	38.4 ± 0.30	39.5 ± 0.46	**37.1** ± 0.41
	# params		-	-	-	333k	407k	445k	268k
Traffic Speed	overall	MAE	4.79	4.03	3.85	3.55 ± 0.01	3.28 ± 0.00	3.10 ± 0.01	**3.05** ± 0.02
		RMSE	8.72	7.94	7.48	7.27 ± 0.01	6.66 ± 0.01	6.31 ± 0.03	**6.25** ± 0.02
	15min	MAE	4.79	3.27	3.16	2.98 ± 0.01	2.83 ± 0.01	2.75 ± 0.01	**2.68** ± 0.02
		RMSE	8.72	6.14	6.05	5.88 ± 0.01	5.47 ± 0.01	5.33 ± 0.02	**5.15** ± 0.02
	30min	MAE	4.79	3.99	3.85	3.57 ± 0.01	3.31 ± 0.00	3.14 ± 0.01	**3.09** ± 0.03
		RMSE	8.72	7.78	7.50	7.26 ± 0.01	6.68 ± 0.00	6.45 ± 0.04	**6.28** ± 0.02
	60min	MAE	4.79	5.18	4.85	4.38 ± 0.01	3.93 ± 0.01	**3.60** ± 0.02	**3.60** ± 0.04
		RMSE	8.72	10.10	9.08	8.88 ± 0.02	8.03 ± 0.02	7.65 ± 0.06	**7.52** ± 0.01
	# params		-	-	-	81k	113k	373k	85k

trainable parameters. However, as the number of parameters shown in Table 3, ST-MetaNet uses less trainable parameters compared with most of deep models. Particularly in traffic speed prediction, ST-MetaNet uses only 23% trainable parameters compared with DCRNN to obtain better results. This fact is related to the good expressiveness of ST-MetaNet that small dimensional hidden states in Meta-GAT and Meta-GRU can have better representation than larger dimensional hidden states in GAT and GRU. Thus the experiments can demonstrate the superiority of ST-MetaNet.

4.3 Evaluation on Meta Networks

To illustrate the effectiveness of each meta learning components, we conduct experiments on five models, including GRU (two GRUs), Meta-GRU (GRU and Meta-GRU), GAT (GRU, GAT, and GRU), Meta-GAT (GRU, Meta-GAT, and GRU), and ST-MetaNet, where the networks in parenthesis denote the layers in Seq2Seq architecture.

As shown in Figure 5, the meta learning based models outperform the conventional models consistently. Specifically, Meta-GRUs are better than GRUs, and Meta-GATs are better than GATs in both tasks. In addition, ST-MetaNet combines both structures of Meta-GRU and Meta-GAT, achieving the best results among all variants. Therefore, the meta-learning-based networks are more effective in modeling ST correlations than conventional network structures.



(a) Taxi flow prediction (b) Traffic speed prediction

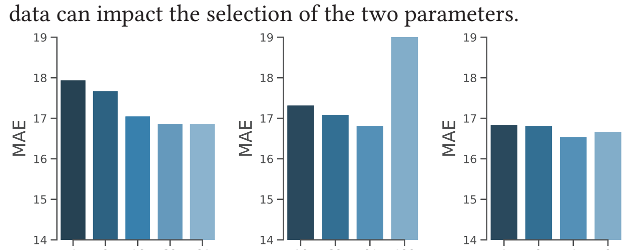
Figure 5: Evaluation on the meta networks

4.4 Evaluation on Framework Settings

ST-MetaNet has many settings, including the number of hidden units within Seq2Seq, the dimension of meta knowledge (outputs of NMK-Learner and EMK-Learner), and the number of hidden units for weight generation. To investigate the robustness of ST-MetaNet, we present the results under various parameter settings.

First, as shown in Figure 6 (a) and Figure 7 (a), increasing the value of meta knowledge dimension enhances the performance significantly in both tasks. As the dimension of meta knowledge does not impact the number of trainable parameters in the Seq2Seq architecture, this fact illustrates that the meta knowledge learned from geo-graph attributes essentially takes effect.

Then in taxi flow prediction task, Figure 6 (b) shows that increasing the number of hidden units in Seq2Seq initially lowers the MAE but then easily overfit when it is too large. While in Figure 6 (c), it presents that the performance is not very sensitive to the number of hidden units for weight generation. In contrast, in traffic speed prediction task, Figure 7 (b) and (c) shows that the performance of traffic speed prediction is very sensitive to the number of hidden units for weight generation, but somehow stable when changing the number of hidden units in Seq2Seq. The reason is that the taxi flow dataset only contains 3600 timestamps, but has 1024 nodes. As a result, less temporal information for Seq2Seq training leads to easily overfitting, but large amounts of nodes make the model effortlessly utilize the geographical information for weight generation. On the contrary, traffic speed prediction task has 34272 timestamps, but only 207 nodes, making it easily overfit when changing the number of hidden units for weight generation, while stable when changing the number of hidden units in Seq2Seq. Overall, the quantity of data can impact the selection of the two parameters.



(a) dimension of meta knowledge (b) # hidden units in Seq2Seq (c) # hidden units for weight generation

Figure 6: Impact of parameters in taxi flow prediction.

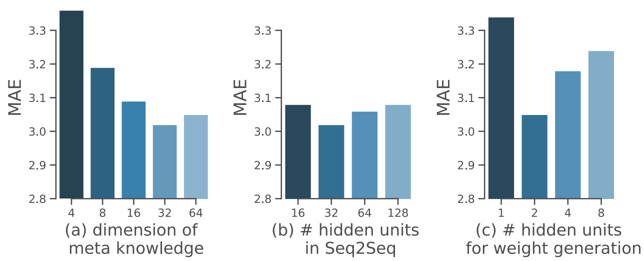


Figure 7: Impact of parameters in traffic speed prediction.

4.5 Evaluation on Meta Knowledge

A good meta learner should obtain the representation of nodes which can reflect traffic similarity. To validate the effectiveness of meta knowledge, for each node we firstly find its k-nearest neighbors in the node embedding space, and then evaluate the similarity of traffic sequences between the node and its neighbors. We employ Pearson correlations and the first order temporal correlations [8], denoted as CORR and CORT respectively, to measure the similarity between two traffic sequences. The similarity functions can be expressed as:

$$\mathrm{CORR}(\mathbf{x},\mathbf{y}) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2}\sqrt{\sum_i (y_i - \bar{y})^2}},$$
$$\mathrm{CORT}(\mathbf{x},\mathbf{y}) = \frac{\sum_i (x_i - x_{i-1})(y_i - y_{i-1})}{\sqrt{\sum_i (x_i - x_{i-1})^2}\sqrt{\sum_i (y_i - y_{i-1})^2}},$$

where $\mathbf{x},\mathbf{y}$ are two temporal sequences, and $\bar{x},\bar{y}$ are their mean values. Note that $-1 \le$ CORR, CORT ≤ 1, and the larger the criteria are, the more similar the two sequences are.

We compare ST-MetaNet with GAT-Seq2Seq, which uses the same Seq2Seq architecture but adopts data fusion strategy to incorporate geographical and traffic information rather than meta learning. As shown in Figure 8, we calculate the traffic similarity on test dataset between each node and its neighbors. The node embeddings of ST-MetaNet in taxi flow prediction task shows significant improvement over embeddings of GAT-Seq2Seq, which implies that ST-MetaNet learns a better geographical representation that reflects the traffic situation. While in traffic prediction task, there is some improvement, but less than the improvement in taxi flow prediction. The reason is that we have much less geographic information in this task. Nonetheless, the result still shows that ST-MetaNet can effectively learn better traffic-related representations.

4.6 Case Study

We further illustrate the effectiveness of ST-MetaNet by performing a case study. A good embedding space should have the characteristic that nearby grids have similar flows. We compare ST-MetaNet with GAT-Seq2Seq by the taxi inflows of three representative grids in Beijing: Yongtai Yuan (residential district), Zhongguancun (business district), and Sanyuan Bridge (viaduct). As shown in Figure 9 (a), the flows of selected grids are obviously distinct from the flows of their neighborhoods produced by GAT-Seq2Seq. While as presented in Figure 9 (b), ST-MetaNet obtains an embedding space that nearby grids have very similar flows. This case demonstrates that ST-MetaNet learns a reasonable representation of nodes, and captures the inherent relationship between geographical information and ST correlations of urban traffic.

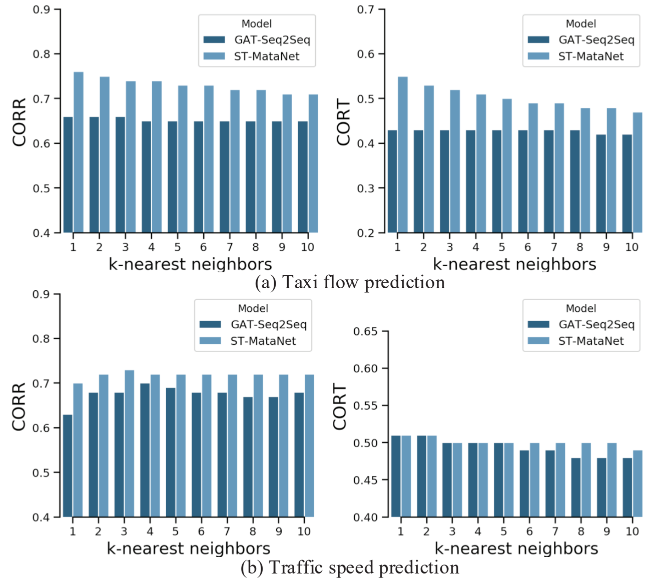


Figure 8: Evaluation on traffic similarity between each node and its k-nearest neighbors in the embedding space.

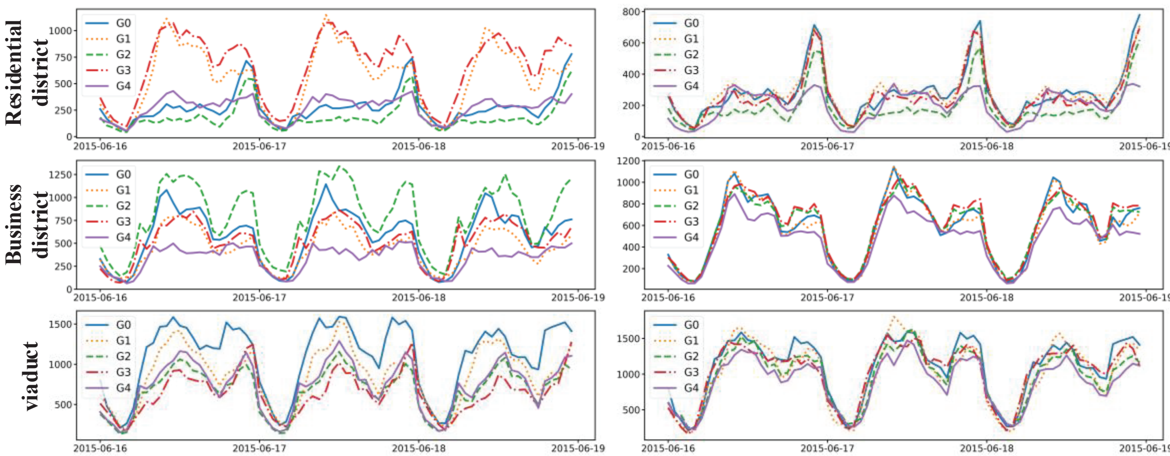
5 RELATED WORK

Urban Traffic Prediction. There are some previously published works on predicting an individual's movement based on their location history [10, 17]. They mainly forecast millions of individuals' mobility traces rather than the aggregated traffic flows in a region. Some other researchers aim to predict travel speed and traffic volumes on roads [1]. Most of them predict single or multiple road segments, rather than citywide ones [6]. Recently, researchers have started to focus on city-scale traffic prediction [14, 27].

Deep Learning for Spatio-Temporal Prediction. Deep learning has powered many application in spatio-temporal areas. In specific, the architectures of CNNs were widely used in grid data modeling like citywide flow prediction [27] and taxi demand inference [24]. Besides, RNNs [9] became popular due to their success in sequence learning, however, these models discard the unique characteristics of spatio-temporal data, *e.g.*, spatial correlation. To tackle this issue, several studies were proposed, such as video prediction [21] and travel time estimation [20]. Very recent studies [7, 15] have indicated that the attention mechanism enables RNNs to capture dynamic spatio-temporal correlation in geo-sensory data.

Meta Learning for Neural Networks' Weight Generation. [3] proposed dynamic filter network to generate convolutional filters conditioned on the input. [4] used a learnet to predict the parameters of a pupil network from a single examplar. [11] employed hypernetworks to generate weights of a large network, which can be regarded as weight sharing across layers. Recently, [5] proposed meta multi-task learning for NLP tasks, which learns task-specific semantic function by a meta network. [26] proposed embedding neural architecture and adopting hypernetwork to generate its weights, to amortize the cost of neural architecture search.

Unlike the above works, we aim to model the diverse ST correlation. To the best of our knowledge, we are the first to study the inherent relationship between geographical information and ST correlations, and apply meta learning on the related application, i.e., urban traffic prediction.



(a) Seq2seq integrated with graph attention (b) The proposed model

Figure 9: The inflows of representative grids. G0 is the selected grid having special function, i.e., residential district, business district, and viaduct. Gk ($k > 0$) denotes the k-nearest neighbors of G0 in the embedding space produced by the models.

6 CONCLUSION AND FUTURE WORK

We proposed a novel deep meta learning framework, named ST-MetaNet, for spatio-temporal data with applications to urban traffic prediction, capable of learning traffic-related embeddings of nodes and edges from geo-graph attributes and modeling both spatial and temporal correlations. We evaluated our ST-MetaNet on two real-world tasks, achieving performance which significantly outperforms 7 baselines. We visualized the similarity of meta-knowledge learned from geographical information, showing the interpretation of ST-MetaNet. In the future, we will extend our framework to a much broader set of urban ST prediction tasks and explore the usage of such representation in other traffic-related tasks.

ACKNOWLEDGMENTS

The work is supported by APEX-MSRA Joint Research Program, and the National Natural Science Foundation of China Grant (61672399, U1609217, 61773324, 61702327, 61772333, and 61632017).

REFERENCES

[1] Afshin Abadi, Tooraj Rajabioun, Petros A Ioannou, et al. 2015. Traffic Flow Prediction for Road Transportation Networks With Limited Traffic Data. *IEEE Trans. Intelligent Transportation Systems* (2015).

[2] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

[3] De Brabandere Bert, Xu Jia, Tinne Tuytelaars, and Luc V Gool. 2016. Dynamic filter networks. In *NIPS*.

[4] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. 2016. Learning feed-forward one-shot learners. In *NIPS*.

[5] Junkun Chen, Xipeng Qiu, Pengfei Liu, and Xuanjing Huang. 2018. Meta Multi-Task Learning for Sequence Modeling. *arXiv preprint arXiv:1802.08969* (2018).

[6] Po-Ta Chen, Feng Chen, and Zhen Qian. 2014. Road traffic congestion monitoring in social media with hinge-loss Markov random fields. In *ICDM*. IEEE.

[7] Weiyu Cheng, Yanyan Shen, Yanmin Zhu, and Linpeng Huang. 2018. A Neural Attention Model for Urban Air Quality Inference: Learning the Weights of Monitoring Stations.. In *AAAI*.

[8] Ahlame Douzal Chouakria and Panduranga Naidu Nagabhushan. 2007. Adaptive dissimilarity index for measuring time series proximity. *ADAC* (2007).

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[10] Zipei Fan, Xuan Song, Ryosuke Shibasaki, and Ryutaro Adachi. 2015. CityMomentum: an online approach for crowd behavior prediction at a citywide level. In *UBICOMP*. ACM.

[11] David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106* (2016).

[12] HV Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M Patel, Raghu Ramakrishnan, and Cyrus Shahabi. 2014. Big data and its technical challenges. *Commun. ACM* (2014).

[13] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[14] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. (2018).

[15] Yuxuan Liang, Songyu Ke, Junbo Zhang, Xiuwen Yi, and Yu Zheng. 2018. GeoMAN: Multi-level Attention Networks for Geo-sensory Time Series Prediction.. In *IJCAI*.

[16] Ye Liu, Yu Zheng, Yuxuan Liang, Shuming Liu, and David S Rosenblum. 2016. Urban water quality prediction based on multi-task multi-view learning. (2016).

[17] Xuan Song, Quanshi Zhang, Yoshihide Sekimoto, and Ryosuke Shibasaki. 2014. Prediction of human emergency behavior and their mobility following large-scale disaster. In *SIGKDD*. ACM.

[18] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

[19] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[20] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks. AAAI.

[21] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and S Yu Philip. 2017. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. In *NIPS*.

[22] Zheng Wang, Kun Fu, and Jieping Ye. 2018. Learning to Estimate the Travel Time. In *SIGKDD '18*.

[23] Jianpeng Xu, Pang-Ning Tan, Lifeng Luo, and Jiayu Zhou. 2016. Gspartan: a geospatio-temporal multi-task learning framework for multi-location prediction. In *SDM*. SIAM.

[24] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, and Jieping Ye. 2018. Deep multi-view spatial-temporal network for taxi demand prediction. (2018).

[25] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. T-drive: driving directions based on taxi trajectories. In *SIGSPATIAL*. ACM.

[26] Chris Zhang, Mengye Ren, and Raquel Urtasun. 2018. Graph HyperNetworks for Neural Architecture Search. *arXiv preprint arXiv:1810.05749* (2018).

[27] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction.. In *AAAI*.

[28] Liang Zhao, Qian Sun, Jieping Ye, Feng Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2015. Multi-task learning for spatio-temporal event forecasting. In *SIGKDD*. ACM.

[29] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: concepts, methodologies, and applications. *TIST* (2014).

[30] Yu Zheng, Yanchi Liu, Jing Yuan, and Xing Xie. 2011. Urban computing with taxicabs. In *UBICOMP*. ACM.

A APPENDIX FOR REPRODUCIBILITY

To support the reproducibility of the results in this paper, we have released our code and data [1]. Our implementation is based on MXNet 1.5.1 [2], and DGL 0.1.3 [3], tested on Ubuntu 16.04 with a GTX 1080 GPU. Here, we detail the datasets, the baseline settings, and the training settings of ST-MetaNet.

A.1 Detailed Settings of Taxi Flow Prediction

Preprocessing of Datasets

We split Beijing city (lower-left GCJ-02 coordinates: 39.83, 116.25; upper-right GCJ-02 coordinates: 40.12, 116.64) into 32×32 grid. The statistics of datasets are in Table 4.

Table 4: Details of the datasets in taxi flow prediction. The datasets tagged by * are processed data for prediction.

Datasets	Property	Value
***Taxi flows**	Timespan	2/1/2015 - 6/2/2015
	Time interval	1 hour
	# timestamps	3600
POIs	# POIs	982,829
	# POI categories	668
Road networks	# roads	690,242
	# road attributes	8
***Geo-graph**	# nodes	1024
	# edges	4114
	# node features	989
	# edge features	32

Taxi flow. We obtain taxi flows from TDrive dataset, which contains a large amount of taxicab trajectories from Feb. 1st 2015 to Jun. 2nd 2015. For each grid, we calculate the hourly inflows and outflows from these trajectories by counting the number of taxis entering or exiting the grid.

POIs. This dataset contains large amounts of POIs in Beijing city. For each grid, we calculate the numbers of POIs for all categories respectively and use them as the node attributes of that grid.

Road networks. The road network dataset contains large amounts of roads in Beijing, each of which has many road attributes, such as length, level, max speed, etc. For each grid, we calculate some road attributes within it, such as the total number of roads and lanes in this grid, as the node attributes. Then if there are roads connecting a pair of grids, we also extract similar road attributes from these roads, as the edge attributes.

In the experiment, we collect geo-graph attributes from POIs and road networks as we demonstrate above. Then we use previous 12-hour flows to predict the next 3-hour flows. We partition the flow dataset on time axis into non-overlapping training, evaluation, and test data, by the ratio of 8:1:1.

Settings of Baselines

The details of the baselines are as follows:

- **HA**. For each grid, we average the hourly flows by dates in the training dataset, which provides 24 inflows and outflows as the prediction results for all days.
- **ARIMA**. We train an individual ARIMA model for each grid and predict future steps separately. The model are implemented using the statsmodel python package. We set the orders as (12,1,0).
- **GBRT**. For each future step (e.g., next 1 hour or next 2 hour), we train a single GBRT and predict the urban traffic, where the input consists of previous traffic information and features of grids. The GBRT is implemented by sklearn python package, with parameters: $n\_estimators = 500$, $max\_depth = 3$, $min\_samples\_split = 2$, and $learning\_rate = 0.01$.
- **Seq2Seq**. We implement a two-layer sequence-to-sequence model for urban traffic prediction, where GRU is chosen as the RNN implementation. It contains 2 layers of GRU, each of which has 128 hidden units. The features of nodes, i.e., node attributes, are firstly embedded by a two-layer FCN with hidden units [32,32]. Then the outputs of FCN fuse with the outputs of the decoder in Seq2Seq. Finally, the fused vectors are linearly projected into two-dimensional vectors, as the predictions of inflow and outflow.
- **GAT-Seq2Seq**. We employ graph attention network and Seq2Seq to model spatial and temporal correlations, respectively. It applies the similar structure as ST-MetaNet, which consists of a GRU layer, a GAT layer, and a GRU layer. All these layers have 128 hidden units, respectively. Like Seq2Seq, we also embed the node attributes by a two-layer FCN with hidden units [32,32], and fuse the embeddings with the outputs of decoder. Finally, the fused vectors are linearly projected into two-dimensional vectors, as the predictions of inflow and outflow.
- **ST-ResNet**. It models the spatio-temporal correlations by ResNet. The ST-ResNet is composed of many residual units with channel size: [64,64,64,64,32,32,32,32,16,16,16,16]. We further fuse its output with embeddings of node attributes, which is encoded by a two-layer FCN with hidden units [32,32]. And finally, we project the hidden states of each grid into two-dimensional vectors, as the predictions of inflow and outflow.

All deep models above are implemented by MXNet.

Settings of ST-MetaNet

The structure of ST-MetaNet for taxi flow prediction is as follow:

- *RNN.* We adopt GRU as the implementation of RNN, in which the dimension of hidden state is 64.
- *NMK-Learner.* It is a two-layer FCN with hidden units [32, 32].
- *EMK-Learner.* It is a two-layer FCN with hidden units [32, 32].
- *Meta-GAT.* The meta learner to generate the weights of GAT is a FCN with hidden units $[16, 2, n]$, where n is the dimension of the target parameters. The Meta-GAT outputs a 64-dimensional hidden states for each grid.
- *Meta-RNN.* We adopt Meta-GRU as Meta-RNN implementation, in which the dimension of hidden state is 64. The meta learner to generate the weights of GRU is a FCN with hidden units $[16, 2, n]$, where n is the dimension of the target parameters.

The Framework is trained by Adam optimizer with learning rate decay. The initial learning rate is 1e-2, and it is divided by 10 every 20 epochs. We also apply gradient clipping before updating the parameters, where the maximum norm of the gradient is set as 5. To tackle the discrepancy between training and inference in Seq2Seq

[1]https://github.com/panzheyi/ST-MetaNet
[2]https://github.com/apache/incubator-mxnet
[3]https://github.com/dmlc/dgl

architecture, we employ inverse sigmoid decay for schedule sampling in the training phase:

$$\epsilon_i = \frac{k}{k + \exp(i/k)}, \tag{4}$$

where we set $k = 2000$. The batch size for taxi flow prediction is set as 16.

A.2 Detailed Settings of Traffic Speed Prediction

Preprocessing of Datasets

In this task, we predict both short-term and long-term traffic speed on road networks. The details of the datasets are as follow.

Traffic speed. We adopt METR-LA dataset, which is collected from loop detectors in the highway of Los Angeles County as shown in Figure 10. This dataset contains traffic speed readings

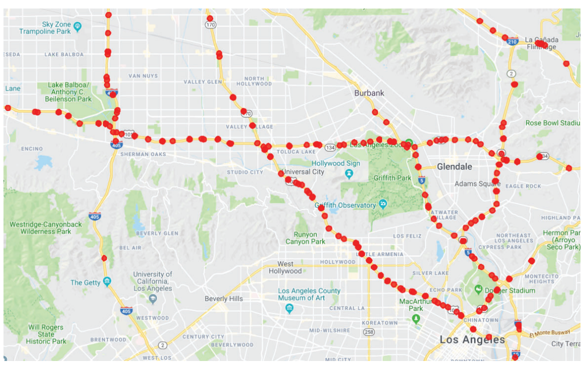


Figure 10: Distribution of loop detectors on roads.

from 207 sensors (nodes). The data is further processed and public in https://github.com/liyaguang/DCRNN, in which the readings are aggregated into 5-minute windows. The details of datasets is shown in Table 5.

Table 5: Details of the datasets in speed prediction. The datasets tagged by * are processed data for prediction.

Datasets	Property	Value
***Traffic speed**	Timespan	3/1/2012 - 6/30/2012
	Time interval	5 minute
	# timestamps	34272
Road networks	# roads	11753
	# road attributes	1
***Geo-graph**	# nodes	207
	# edges	3726
	# node features	20
	# edge features	1

Road networks. This dataset only contains road distance between nodes. We firstly extract road structure information for each node as the node attributes. Specifically, the road structure for a single node is a collection of information about the distance between the node and its k-nearest neighbors. For each node, we sort the k-nearest distance, and get the sorted array as the attributes of this node. Then we extract edge attributes, which are simply defined as the road distance between nodes. For the efficiency of model training and testing, we only keep the edge between each node and its k-nearest neighbors. Since the traffic correlations are directional on road networks, we collect edge attributes in both directions respectively.

In addition, the GPS coordinates of all loop detectors, are also used as the node attributes.

In this task, we set $k = 8$, and get the geo-graph data from road networks as we demonstrate above (in Table 5). We use previous 60-minute traffic speed to predict speed in the next 60 minutes. We partition the traffic speed dataset on time axis into non-overlapping training, evaluation, and test data, by the ratio of 7:1:2.

Settings of Baselines

The details of the baselines are as follows:

- **HA**. For each grid, we average the 5-minute speed by dates in the training dataset, which provides 24×12 speed readings as the prediction results for all days.
- **ARIMA**. We train an individual ARIMA model for each loop detectors. We set the orders as (12,1,0).
- **GBRT**. We employ the same model as the GBRT in taxi flow prediction, with parameters: $n\_estimators = 500$, $max\_depth = 3$, $min\_samples\_split = 2$, and $learning\_rate = 0.01$.
- **Seq2Seq**. We employ the same Seq2Seq model as taxi flow prediction. The hidden units of GRUs are 64, and the node attributes are embedded by a two-layer FCN with hidden units [32,32].
- **GAT-Seq2Seq**. We employ the same GAT-Seq2Seq model as taxi flow prediction. The hidden units of GAT and GRUs are 64, and the node attributes are embedded by a two-layer FCN with hidden units [32,32].
- **DCRNN**. This model is state-of-the-art in traffic prediction on road networks. it employs the diffusion convolution operator and Seq2Seq to capture spatial and temporal correlations, respectively. We use the implementation of DCRNN from its author's github [4] (*version 80e156c*). We apply a two-layer Seq2Seq network based on diffusion convolution gated recurrent unit with 64 hidden units to predict future traffic.

Settings of ST-MetaNet

The structure of ST-MetaNet for taxi flow prediction is as follow:

- *RNN*. We adopt GRU as the implementation of RNN. The dimension of its hidden state is 32.
- *NMK-Learner*. It is a two-layer FCN with hidden units $[32, 32]$.
- *EMK-Learner*. It is a two-layer FCN with hidden units $[32, 32]$.
- *Meta-GAT*. The meta learner to generate the weights of GAT is a FCN with hidden units $[16, 2, n]$, where n is the dimension of the target parameters. The Meta-GAT outputs a 32-dimensional hidden states for each grid.
- *Meta-RNN*. We adopt Meta-GRU as Meta-RNN implementation, in which the dimension of hidden state is 32. The meta learner to generate the weights of GRU is a FCN with hidden units $[16, 2, n]$, where n is the dimension of the target parameters.

The Framework is trained by Adam optimizer with learning rate decay. The initial learning rate is 1e-2, and it is divided by 10 every 20 epochs. We also apply gradient clipping with the maximum norm of gradient 5. We employ inverse sigmoid decay for schedule sampling in the training phase (as Eq. 4), where we set $k = 2000$. The batch size for traffic speed prediction is set as 32.

[4] https://github.com/liyaguang/DCRNN