Meta-STMF: Meta-Learning Based Spatial Temporal Prediction Model Fusion Approach

LIU WEI AND JIA SULING
School of Economics and Management, Beihang University, Beijing 100191, China
Corresponding author: Jia Suling (jiasuling@buaa.edu.cn)

ABSTRACT Traffic prediction is one of the core technology of intelligent transportation systems and is of great importance in traffic management. Among the existing methods, deep learning-based traffic prediction models are the mainstream methods and have achieved decent performance. However, when faced with traffic data with diverse traffic patterns, one model has to divide its attention to account for the prediction performance in different patterns, thus degrading the overall performance. To overcome the limitation, this paper proposes a meta-learning-based spatial-temporal deep learning model fusion approach, called Meta-STMF. The main idea is to assemble a group of sub-models so that each sub-model can focus on the patterns it specializes in. The method first trains each sub-model separately. Then, it extracts meta-knowledge from the input traffic data and utilizes a meta-learner to generate adaptive combination weights for each sub-model. Finally, the combination weights are used in the fusion of every sub-model prediction to obtain the final prediction. Extensive experiments on two large-scale real-world datasets show that our proposed approach consistently outperforms all competing baseline models with a large margin.

INDEX TERMS Traffic prediction, meta-learning, model fusion.

I. INTRODUCTION
Rapid urbanization has brought about the growth of the urban population and the number of vehicles, which presented huge traffic congestion challenges to modern cities. The Intelligent Transportation System (ITS) has received increasing attention because of its potential to assist traffic management and alleviate traffic congestion. As the core technology of the ITS, traffic prediction aims to predict the future traffic conditions given historical traffic observations. Accurate traffic prediction can help traffic management better allocate and dispatch vehicles in advance to avoid traffic congestion, such as implementing traffic control for road segments (or regions) predicted to have high traffic flow. On the other hand, the prediction results can assist drivers to choose a smoother road with fewer vehicles to travel, thus alleviating traffic congestion.

Many efforts have been conducted to develop methods for the traffic prediction problem [1], [2]. Early approaches treat the problem as a time series analysis and employ traditional statistical models such as auto-regressive integrated moving average (ARIMA) [3] and Kalman filtering [4]. These models usually rely on the stationary assumption, which is often violated by traffic data. Recently, deep learning models for traffic prediction have been developed and achieved state-of-the-art performance. These methods usually capture dynamic temporal correlations through recurrent neural networks [5] or temporal convolutional networks [6] and extract spatial dependencies by standard convolutional neural networks (for grid-based data) [7] or graph neural networks (for graph-based data) [8]. Due to the powerful ability to capture spatial and temporal characteristics from traffic data [9], [10], the deep learning-based spatial-temporal prediction models become the mainstream method for traffic prediction.

Although deep learning-based models have achieved decent performance, they cannot well model traffic data with diverse traffic patterns. For example, the traffic patterns of a transportation hub and a residential area are quite different. If we simply employ a single model to capture both patterns, the model has to divide its attention to account for the prediction performance in different patterns. This limits the potential of the model and we can only get the degraded
prediction performance. This situation becomes worse when the patterns are more diverse and complex.

To tackle this challenge, we propose a Meta-learning-based Spatio-Temporal deep learning Model Fusion approach, called Meta-STMF. Our main idea is to assemble a group of sub-models so that each sub-model can focus on the patterns it specializes in, thus better modeling traffic data with diverse traffic patterns. Firstly, we separately train each sub-model in the model pool on the traffic prediction task. Secondly, we manually extract multiple features with temporal and spatial characteristics as meta-knowledge from each input traffic time series and employ a meta-learner to generate adaptive combination weights of sub-models for each traffic time series based on its meta-knowledge. Finally, we assemble prediction results of the well-trained sub-models using the combination weights to obtain the final prediction. Our approach trains the model in a two-stage manner, where the first training stage is the first step and the second training stage includes the last two steps. The effectiveness of the proposed method is verified by extensive experiments over two large-scale real-world traffic datasets including LOS_LOOP and PEMSD4. In summary, our main contributions are three-fold:

- We propose a meta-learning-based model fusion approach (Meta-STMF) for spatio-temporal deep learning models to effectively model traffic data with diverse traffic patterns in an ensemble manner.
- We design an adaptive combination weight generation strategy, which utilizes a meta-learner to dynamically generate meta-knowledge-based weights for each traffic time series. The meta-knowledge is manually extracted from traffic series data.
- Extensive experiments on two large-scale real-world traffic datasets show that our model outperforms eight competing baseline methods including the state-of-the-art method.

The outline of this paper is as follows. Section II analyzes the related work and the difference between our approach and existing methods. Section III gives basic concepts and a formal problem definition. Section IV details our proposed Meta-STMF. Section V evaluates the proposed approach on two real-world datasets. The last section summarizes this paper and looks forward to future work.

II. RELATED WORK

In this section, we introduce the related work of this paper, which lies in two aspects including traffic prediction and meta-learning. Then, we discuss the main differences between this paper and the related work.

A. TRAFFIC PREDICTION

Traffic prediction refers to predicting future traffic conditions based on historical traffic data. The traffic conditions can be described mainly by traffic speed and traffic flow, thus resulting in two tasks traffic speed prediction and traffic flow prediction. Traffic speed prediction is to predict the average speed of vehicles on the road for a period of time in the future. Traffic flow prediction refers to forecasting the number of vehicles entering or leaving a certain area or road section within a period of time in the future. Some studies also predict the flow of transfer between different locations.

Traffic prediction is treated as a time series forecasting problem. The time-series community usually employs statistical models such as ARIMA and Kalman filtering. However, there is a gap between time series forecasting and traffic prediction. The key difference is that traffic data are highly dynamic in temporal dependency and have a strong spatial correlation. Therefore, modern deep learning-based methods usually consider both aspects and achieve advanced performance. Details will be introduced next.

1) SPATIAL CORRELATIONS

In terms of spatial correlation modeling, widely used methods include convolutional neural networks (CNN) and graph neural networks (GNN), especially graph convolutional networks (GCN) [11] and graph attention networks (GAT) [12].

Since the traditional CNN model can only model Euclidean data, researchers usually convert the traffic network structure at different times into the form of “images” and divide it into a standard grid structure according to latitude and longitude. Many CNN-based models use each grid to represent a region, and then use CNN to learn the spatial correlation between different regions, such as ST-ResNet [13].

The real urban transportation network is more similar to the graph structure. The GCN is suitable for this structure, so most of the current research methods use GCN to directly model the traffic data of the graph structure. Since 2018, almost every newly proposed traffic prediction model is inseparable from graph convolution. Graph convolution includes spectrum-based methods and space-based methods, both of which have achieved good results in traffic prediction. A key problem of the graph convolution model is the construction of the graph adjacency matrix. The simplest method is to treat traffic sensors deployed on roads of the urban road network as nodes in the graph, and the distance between the sensors as the weight of the edges in the graph, such as DCRNN [14], STGCN [15], TGCN [16]. Some studies have also proposed an adaptive graph generation module because the graph adjacency matrix using a fixed algorithm (or distance or similarity) is quite intuitive and may be incomplete. The graph adjacency matrix calculated according to the predetermined algorithm may not necessarily reflect the spatial correlation of real traffic data, and the adaptive method can automatically learn this spatial correlation from the data, such as Graph-WaveNet [6], MTGNN [17], HGCN [8].

The attention mechanism also has a good application in the spatial correlation modeling of traffic prediction. The traffic status of a road will be affected by the traffic status of other roads, but it does not mean that the closer the distance in space, the greater the impact. Because this “influence” is not static, but generally changes dynamically over time. The spatial attention is to assign different weights to different positions, so that the model can autonomously learn the
dynamic spatial correlation contained in the traffic data, such as ASTGCN [18], GMAN [19], STAGGCN [20].

2) TEMPORAL DEPENDENCIES
In the temporal dependency modeling of traffic prediction, widely used methods include recurrent neural network (RNN), 1D-CNN, attention mechanism, and so on.

Due to the powerful ability of RNN in modeling sequence data, RNN is often used to model the time correlation in traffic data in previous studies, such as TGCN [16], TGCLSTM [21]. RNN has also been extended to an “encoder-decoder” structure. This structure can encode the input traffic data sequence to generate an internal representation vector and then use the decoder to generate the prediction results of multiple time steps. This structure fits the characteristics of “multi-step forecasting” of traffic prediction, so it has been widely used in the field of traffic prediction, such as DCRNN [14], AGCRN [22].

1D-CNN has advantages in time correlation modeling because it does not depend on the result of the previous time step. Therefore, parallel computing can be realized and it is more efficient. Some traffic prediction models use CNN to build time convolution to model time correlation. These studies generally use 1D convolution in the time dimension to capture time correlation, such as STGCN [15], ASTGCN [18]. Some models introduce dilated causal convolution, where the receptive field of each layer can be expanded exponentially so that the expanded time convolution module can capture longer (further) time dependence, such as GraphWaveNet [6], MTGNN [17].

Similar to the spatial attention mechanism, the temporal attention mechanism can also adaptively learn the non-linear time correlation at different time steps by assigning different weights to the traffic state at different time steps, such as ASTGCN [18], GMAN [19].

Although many efforts have been devoted to traffic prediction, they mainly focus on utilizing one single model to make predictions. This limits model’s prediction performance when faced with data with diverse traffic patterns. Inspired by the ensemble model, our approach leverages a model pool (containing several models) and a meta-learning-based model fusion strategy to make accurate predictions.

B. META LEARNING
Meta-learning is well known as learning to learn. The goal of meta-learning is to train a model that can quickly adapt to a new task using only a few data points and training iterations. There are many implementations of meta-learning, one of which is optimization-based meta-learning [23]. In this type of meta-learning, there are two key concepts: the meta-learner and the learner. The original model for handling the task is called “learner”, while the goal of the meta-learner is to efficiently update the learner’s parameters so that the learner can adapt to the new task quickly. In our paper, the meta-learner weights the output of multiple traffic prediction models to make predictions. This set of weighting coefficients is generated by the meta-learner based on the intrinsic feature of the data, i.e., meta-knowledge.

For time series model fusion, a common meta-learning approach is to select the best sub-model in the pool of sub-models for each series, i.e., the model that produces the lowest forecast loss. This approach treats the problem as a classification problem by setting the individual sub-models as the classes and the best sub-model as the target class. However, there may be other sub-models that produce similar prediction errors to the best sub-model, so the specific class chosen is less important than the prediction error produced by each sub-model. Therefore, we transform the problem into a combination weight generation problem that assigns weights for the prediction results of each sub-model to obtain the final prediction. Also, our method can be considered a soft classification.

III. PRELIMINARIES
In this section, we introduce some basic concepts and give a formal definition of our problem.

A. BASIC CONCEPTS
Definition 1 (Road Network): A Road Network is a graph \( G = (V, E, A) \), where \( V \) is the set of all different sensor nodes, \( \mid V \mid = N \), \( E \) is the set of edges, and \( A \in R^{N \times N} \) is the weighted adjacency matrix. It can express whether two entities are connected or not and the weight of the link.

Definition 2 (Traffic Data Matrix): A Traffic Data Matrix is composed of the features of each node on the road network, expressed as \( X \in R^{N \times H} \). \( H \) represents the length of the time series. Therefore, the traffic data matrix \( X \) can be seen as a combination of \( H \) time steps of graph signals, or as a combination of \( N \) time series. It is worth noting that the features of this node can be any traffic information, such as traffic speed, traffic flow, traffic density, and so on.

Definition 3 (Model Pool): A Model Pool is a pool of sub-models. As a model fusion approach, our method Meta-STMF is based on a group of sub-models. Here we use \( M_p \) to represent our model pool including some selected spatial-temporal prediction models, where the total number of sub-models is \( M \). Each sub-model can independently complete the traffic prediction task and produce a prediction result.

B. PROBLEM DEFINITION
Traffic prediction task refers to forecasting future traffic conditions based on historical traffic data, defined as:

\[ \hat{O} = \mathcal{M}(G, X), \]  

where \( G \) is the road network, \( X \in R^{N \times T} \) includes the traffic data from time step \( t - T + 1 \) to \( t \), and \( \hat{O} \in R^{N \times T'} \) denotes the predicted traffic conditions from time step \( t + 1 \) to \( t + T' \). \( \mathcal{M} \) is the learned mapping function for traffic prediction.

Given the road network structure \( G \), the traffic data matrix \( X \in R^{N \times T} \), and the model pool \( M_p \), the goal of our spatial-temporal model fusion approach, Meta-STMF, is to learn an
meta-learner \( \text{MetaF} \) to fuse the prediction results of all sub-models in the model pool \( M_p \) on the traffic matrix \( X \). It can be defined as:

\[ \hat{Y} = \text{MetaF}(M_1(G, X), M_2(G, X), \ldots, M_M(G, X)) \]  

where \( \hat{Y} \in \mathbb{R}^{N \times T'} \) is the final output of our approach.

### IV. METHODOLOGY

In this section, we first describe the model pool used in Meta-STMF. Then, we introduce the meta-learner and detail the meta-knowledge extraction and combination weight generation. Finally, we present the model fusion and the training strategy of our Meta-STMF. The overall framework of our Meta-STMF is shown in Figure 1.

#### A. MODEL POOL

Due to the powerful modeling ability, we only consider deep learning models for traffic prediction as sub-models in our model pool. To guarantee the diversity of the model pool, we account for a wide range of traffic prediction models, including traditional ones: (i) fully-connected gated recurrent unit (GRU), (ii) sequence-to-sequence model with GRU core (Seq2Seq), (iii) auto-encoder model (AutoEncoder), and modern ones: (i) attention-based spatial-temporal graph convolutional network (ASTGCN) [18], (ii) CONVGCN [24], (iii) ATDM [25], (iv) STTN [26]. These models are also used as baseline methods with details described in the experiment section.

Given a model pool, we train the \( m \)-th sub-model on the input traffic data and obtain the output of the sub-model \( \hat{O}_m \in \mathbb{R}^{N \times T'} \) through Eq. (1). For the traffic time series of graph node \( i \), the output of the \( m \)-th model is denoted as \( \hat{o}_{im} \in \mathbb{R}^{T'} \).

#### B. META-LEARNER

In this section, we first describe the extraction of meta-knowledge from each traffic time series. Then, we introduce the meta-learner that aims to generate combination weights of sub-models prediction results based on the meta-knowledge.

1) META-KNOWLEDGE EXTRACTION

We extracted several features as meta-knowledge from the input traffic data. The meta-knowledge covers not only common features of time series but also features involving spatial information of graph nodes. The detailed meanings of the characteristics are given in the appendix. Moreover, the meta-knowledge varies with each node’s traffic series in this paper because different graph nodes usually have different traffic patterns. For traffic time series of node \( i \), we extracted \( P \) features and identified them with \( f_i \in \mathbb{R}^P \).

2) COMBINATION WEIGHT GENERATION

Based on the meta-knowledge, we construct our meta-learner to assign combination weight for each sub-model. Specifically, we employ an Xgboost model [27] as our meta-learner, which belongs to the gradient boosting method. The formal expression is

\[ p_i = \text{ML}(f_i), \]  

where \( \text{ML} \) represents the meta-learner, \( f_i \in \mathbb{R}^P \) is the input meta-knowledge of node \( i \)’s traffic series and \( p_i \in \mathbb{R}^M \) denotes the output of meta-learner.

We do not directly use the \( p_i \) as the combination weights for sub-models, but first pass them through a Softmax function so that the sum of the weights is 1. The process is defined as:

\[ w_{im} = \frac{\exp(p_{im})}{\sum_{m=1}^{M} \exp(p_{im})}, \]  

where \( w_{im} \in \mathbb{R} \) is the weight of \( m \)-th sub-model for the traffic series of graph node \( i \).

3) META-LEARNER OPTIMIZATION

Our meta-learner is an integrated learning approach that uses multiple weakly learning decision trees \( T_k, k = 1, 2, \ldots, K \) to form a strong learning decision tree model \( F_K \).
sequentialy:

\[
\mathcal{F}_K(x) = \sum_{k=1}^{K} \mathcal{T}_k(x),
\]

where \( \mathcal{T}_k(x) \) is the prediction score of the k-th decision tree on input \( x \), and \( \mathcal{F}_K \) is the final prediction score of the meta-learner. From Eq. (5), we can derive that \( \mathcal{F}_K(x) = \mathcal{F}_{K-1}(x) + \mathcal{T}_k(x) \), which is very helpful in solving for \( \mathcal{T}_k(x) \).

Given the loss function \( \mathcal{L} \), we know that \( \mathcal{L}(\mathcal{F}_K(x)) = \mathcal{L}(\mathcal{F}_{K-1}(x) + \mathcal{T}_k(x)) \). The Taylor expansion approximation leads to

\[
\mathcal{L}(\mathcal{F}_K(x)) \approx \left[ \mathcal{L}(\mathcal{F}_{K-1}(x)) + g \cdot \mathcal{T}_k(x) + \frac{h \cdot \mathcal{T}_k(x)^2}{2} \right],
\]

where \( g = \frac{\partial \mathcal{L}(\mathcal{F}_{K-1}(x))}{\partial \mathcal{F}_{K-1}(x)} \), \( h = \frac{\partial^2 \mathcal{L}(\mathcal{F}_{K-1}(x))}{\partial \mathcal{F}_{K-1}(x)^2} \).

According to Eq. (6), the partial derivative of \( \mathcal{L}(\mathcal{F}_K(x)) \) with respect to the \( K \)-th decision tree \( \mathcal{T}_k(x) \) is

\[
\frac{\partial \mathcal{L}(\mathcal{F}_K(x))}{\partial \mathcal{T}_k(x)} \approx \frac{\partial \mathcal{L}(\mathcal{F}_{K-1}(x) + \mathcal{T}_k(x))}{\partial \mathcal{T}_k(x)} \approx g + h \cdot \mathcal{T}_k(x).
\]

To minimize the loss function, we make Eq. (7) equals to zero and obtain:

\[
\mathcal{T}_k(x) = \frac{g(x)}{h(x)}.
\]

Therefore, the key to optimize our meta-learner is to compute \( g \), \( h \) and we will give the exact formula in the next section.

C. MODEL FUSION AND TRAINING

For node \( i \)'s traffic series, we have obtained the combination weight of \( M \) sub-models, so that we can combine the output of each sub-model (\( \tilde{o}_{im} \in \mathbb{R}^T \)) to calculate the final prediction result as follows:

\[
\hat{y}_i = \sum_{m=1}^{M} w_{im} \tilde{o}_{im},
\]

where \( \hat{y}_i \in \mathbb{R}^T \). Evaluating the error between the prediction and the ground truth data, we can define the loss function as follows:

\[
\mathcal{L} = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T'} (\hat{y}_{it} - y_{it})^2,
\]

where \( \hat{y}_{it} \) denotes the predicted value for node \( i \)'s traffic series at \( t \)-th time step and \( y_{it} \) is the ground truth.

Recall that \( g \), \( h \) in Eq. (8) is important to the optimization of our meta-learner. We give the exact formula of them based on \( \mathcal{L}_i \), which is the loss of node \( i \)'s traffic series:

\[
g_{im} = \frac{\partial \mathcal{L}_i}{\partial p_{im}} = 2 w_{im} \sum_{t=1}^{T'} (\hat{y}_{it} - y_{it})(\hat{o}_{it} - \hat{y}_{it}),
\]

\[
h_{im} = \frac{\partial^2 \mathcal{L}_i}{\partial p_{im}^2} = (1 - 2w_{im})g_{im} + 2w_{im}^2 \sum_{t=1}^{T'} (\hat{o}_{it} - \hat{y}_{it})^2,
\]

where \( g_{im}, h_{im} \) is the \( m \)-th component of the first-order and second-order gradient of the objective function on the \( i \)-th series, \( \hat{o}_{it} \) is the prediction of the \( m \)-th model at the \( t \)-th time step for the \( i \)-th series.

V. EXPERIMENTS

In this section, we evaluate our proposed method on two large-scale real-world traffic datasets.

A. DATASETS

The two real-world public traffic datasets in our paper include LOS_LOOP dataset and PEMSD4 dataset, which is detailed as follows:

- **LOS_LOOP**: This dataset contains real-time traffic speed data collected by 207 loop detectors deployed on highways in Los Angeles County. The data collection time is from March 1st to March 7th, 2012, and we aggregated the traffic speed every 5 minutes. This dataset also provides the adjacency matrix of the road network, which is calculated by the distance between the sensors in the traffic network. Since this dataset contains some missing data, we use linear interpolation to fill in the missing values.

- **PEMSD4**: The PeMSD4 dataset refers to the traffic flow data in the San Francisco Bay Area. There are 307 loop detectors operated within the period from 1/Jan/2018 to 28/Feb/2018. This dataset also includes the adjacency matrix between the nodes. The time interval for collecting traffic speed data in this dataset is 5 minutes. We take the traffic data from it in the first week (1/Jan/2018 to 7/Jan/2018) for experiments.

B. BASELINES

We compare the performance of the Meta-STMF model with the following baseline methods, including one traditional method and seven deep learning-based approaches:

- **HA**: Historical Average, which models the historical traffic as a seasonal process, then uses the weighted average of previous seasons as predicted values. Here we use the data of the past 4 weeks to make predictions. Because the historical average method does not rely on short-term data, its performance is unchanged for small increases in the forecast horizon.

- **GRU**: The baseline model is implemented by ourselves for traffic state prediction task based on gated recurrent unit (GRU) and a full-connected network as the output layer.

- **Seq2Seq**: We utilize the sequence-to-sequence framework based on recurrent neural network, i.e., GRU, for multi-step prediction.

- **AutoEncoder**: This baseline model uses an encoder to learn an embedded vector from data and then employs a decoder to predict the future traffic state.

- **ASTGCN** [18]: Attention-based spatio-temporal graph convolutional network, which combines the spatial-temporal
TABLE 1. Model comparison on metrics MAE/MAPE/RMSE, where MAPE is in %. Our Meta-STMF significantly outperforms all competing baselines with regard to all metrics over all datasets according to student’s t-test at level 0.01.

| Dataset | Method     | Step 1 MAE | MAPE(%) | RMSE  | Step 2 MAE | MAPE(%) | RMSE  | Step 3 MAE | MAPE(%) | RMSE  |
|---------|------------|------------|---------|--------|------------|---------|--------|------------|---------|--------|
| LOS. LOOP | HA         | 6.120      | 17.550  | 10.809 | 6.120      | 17.550  | 10.809 | 6.120      | 17.550  | 10.809 |
|         | GRU        | 3.963      | 10.322  | 6.452  | 4.007      | 10.496  | 6.656  | 4.049      | 10.630  | 6.816  |
|         | Seq2Seq    | 3.874      | 10.829  | 6.949  | 3.895      | 10.911  | 7.039  | 3.947      | 11.038  | 7.162  |
|         | AutoEncoder| 3.878      | 11.002  | 6.904  | 3.907      | 11.046  | 6.989  | 3.938      | 11.124  | 7.093  |
|         | ASTGCN     | 3.744      | 10.744  | 6.464  | 3.833      | 10.777  | 6.657  | 4.334      | 10.977  | 6.838  |
|         | CONVGCN    | 3.757      | 10.646  | 6.515  | 3.887      | 10.810  | 6.598  | 3.941      | 11.000  | 6.726  |
|         | ATDM       | 3.785      | 10.452  | 6.505  | 3.823      | 10.582  | 6.564  | 3.871      | 10.710  | 6.699  |
|         | STTN       | 3.562      | 9.745   | 6.435  | 3.628      | 9.995   | 6.602  | 3.709      | 10.303  | 6.824  |
|         | Meta-STMF  | 3.363      | 9.327   | 5.798  | 3.443      | 9.603   | 5.973  | 3.522      | 9.869   | 6.163  |
|         | Improvement| 5.582%     | 4.290%  | 9.892% | 5.093%     | 3.914%  | 9.283% | 5.053%     | 4.213%  | 9.998% |

C. EVALUATION METRICS

To describe the prediction performance, we use three metrics to evaluate the difference between the real traffic conditions and the predicted value, including the mean absolute error (MAE), the root mean square error (RMSE), and the mean absolute percentage error (MAPE).

- Mean Absolute Error (MAE):

\[ \text{MAE}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \]  \hspace{1cm} (13)

- Root Mean Square Error (RMSE):

\[ \text{RMSE}(y, \hat{y}) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \]  \hspace{1cm} (14)

- Mean Absolute Percentage Error (MAPE):

\[ \text{MAPE}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100\% \]  \hspace{1cm} (15)

D. EXPERIMENT SETTINGS

In the experiments, Z-Score normalization was used for the input data. In addition, in chronological order, 70% of the data is selected as the training set, 10% as the validation set, and the remaining 20% as the test set. For multi-step prediction, in the experiments of the two datasets, the historical data of 12 timesteps are used to predict the data of 3 timesteps in the future. The data batch size is set to 64, all models are trained for 100 epochs. We stop the training process if the loss on the validation set persists for 50 epochs without decreasing.

For Meta-STMF, we use the Bayesian Optimization Approach [28] for hyper-parameter search. After search, the parameters of the meta-learner (Xgboost) are max_depth: 17, eta: 0.4752, subsample: 0.4971, colsample_bytree: 0.6249, tree_method: exact. In the training process of meta-learner, we employ the training and validation sets used in the sub-model training (80% of the data) as the training set, and the test set is the same as the sub-model’s (the remaining 20% of the data).
E. OVERALL RESULTS

Table 1 shows the comparison results of our Meta-STMF and eight baseline models on the LOS_LOOP dataset and the PEMSD4 dataset using historical data of 12 timesteps to predict the next 3 timesteps.

We can observe the following phenomena. First of all, traditional method HA performs poorly because they cannot handle the complex nonlinear correlations in spatial-temporal traffic data.

Secondly, spatial-temporal methods based on graph neural networks, including ASTGCN, CONVGCN, ATDM, and STTN, are superior to shallow neural networks only consider time correlations, including GRU and Seq2Seq. This illustrates the importance of considering the spatial correlation of the road network structure in traffic prediction.

Additionally, our Meta-STMF has significantly improved the performance of the spatial-temporal prediction methods and obtained the best performance over all prediction horizons and all metrics except MAPE for PEMSD4. Compared with other methods on the LOS_LOOP dataset, it has 4%-10% improvement, and there is an improvement of 2%-5% on the PEMSD4 dataset, which shows the effectiveness of our proposed meta-learning based adaptive fusion method.

F. ABLATION STUDY

To verify the effectiveness of our proposed meta-learning-based adaptive model fusion method, we compared with two variants as an ablation study:

- **Meta-STMF-same-weight**: This is a variant of our proposed model where we set the model weighting parameter $w_{im}$ mentioned above to $1/M$, which means that the same weights are applied for all sub-models to get the integrated prediction results.

- **Meta-STMF-classification**: This is a variant of our proposed model that removes the weighted fusion part. Instead, Meta-STMF-classification treats the model fusion problem as a model classification problem and chooses the best model for each traffic series.

The results is showed at Figure 2. From the figure, we can see Meta-STMF-classification performs the worst, because it cannot leverage more information from different sub-models. Different from it, Meta-STMF and Meta-STMF-same-weight transform the model fusion problem as a combination weight assigning problem and utilize abundant information learned by different sub-models, thus obtaining better performance. However, our Meta-STMF outperforms the Meta-STMF-same-weight that is a simple same weighted fusion method and ignore the performance difference of different sub-models for different traffic series. The goal of our meta-learning-based model is to learn a combination weight for each sub-model in the model pool and calculate the multi-model fusion results as the final prediction. Overall, the ablation study shows that the meta-learning based adaptive model average method is better than the arithmetic average (such as Meta-STMF-same-weight).

VI. CONCLUSION AND FUTURE WORK

This paper proposes a meta-learning-based spatial-temporal deep learning model fusion approach, called Meta-STMF. The approach adaptively generates the weighted traffic prediction results of a set of deep learning models. The weight generation takes the meta-knowledge with temporal and spatial characteristics extracted from the input traffic data. Experiments show that our Meta-STMF outperforms all baseline models with a large margin.

However, there are some limitations to this work. For instance, we train the meta-learner by some artificial features, and these may not fully cover the spatio-temporal properties in the data. In future work, we will try to deepen the model and use the powerful representational power of deep learning models such as graph convolutional networks (GCNs) [11] to automatically extract spatio-temporal features from the data for model fusion. And, since our scheme is a general model fusion scheme, it is not limited to the traffic domain. Therefore, we will introduce more datasets and baseline models from other domains to enrich the application scenarios and values of our proposed scheme.

APPENDIX

SPATIAL-TEMPORAL META-KNOWLEDGE

In this appendix, we list all the features extracted from traffic data, which is used as our meta-knowledge to generate combination weights. They are detailed as follows:

- degree: The degree of each node
- $x_{acf}$: The first autocorrelation coefficient of the series
• x_acf10: The sum of the squared first ten autocorrelation coefficients of the series.
• diff1_acf1: The first autocorrelation coefficient of the first differenced series
• diff1_acf10: The sum of the squared first ten autocorrelation coefficients of the first differenced series.
• diff2_acf1: The first autocorrelation coefficient of the twice-differenced series.
• diff2_acf10: The sum of squared fist ten autocorrelation coefficients of the original series.
• seas_acf1: The autocorrelation coefficient at the first seasonal lag. If the series is non seasonal, this feature is set to 0.
• arch_lm: A statistic based on the Lagrange Multiplier test of Engle (1982) for autoregressive conditional heteroscedasticity. The $R^2$ of an autoregressive model of 12 lags applied to $x^2$ after the its mean has been subtracted.
• crossing_point: The number of times the series crosses the median.
• entropy: The spectral entropy of the series. $H_2(x_t) = -\int_{-\pi}^{\pi} f_x(\lambda) \log f_x(\lambda) d\lambda$ where the density is normalized so $\int_{-\pi}^{\pi} f_x(\lambda) d\lambda = 1$
• flat_spots: The number of flat spots in the series, calculated by discretizing the series into 10 equal sized intervals and counting the maximum run length within any single interval.
• arch_acf: After the series is pre-whitened using an AR model and squared, the sum of squares of the first 12 autocorrelations.
• garch_acf: After the series is pre-whitened using an AR model, a GARCH(1,1) model is fitted to it and the residuals are calculated. The sum of squares of the first 12 autocorrelations of the squared residuals.
• garch_r2: After the series is pre-whitened using an AR model, a GARCH(1,1) model is fitted to it and the residuals are calculated. The sum of squares of the first 12 autocorrelations of the squared residuals.
• alpha $\alpha$: The smoothing parameter for the level in an ets(A,A,A) model fitted to the series.
• beta $\beta$: The smoothing parameter for the trend in a ets(A,A,A) model fitted to de series.
• hurst: The hurst coefficient indicating the level of fractional differencing of a time series.
• lumpiness: The variance of the variances based on a division of the series in non-overlapping portions. The size of the portions if the frequency of the series, or 10 is the series has frequency 1.
• nonlinearity: A nonlinearity statistic based on Terasvirta’s nonlinearity test of a time series.
• x_pacf5: The sum of squared first 5 partial autocorrelation coefficients of the series.
• diff1x_pacf5: The sum of squared first 5 partial autocorrelation coefficients of the first differenced series.
• seas_pacf: The partial autocorrelation coefficient at the first seasonal lag. 0 if the series is non seasonal.
• nperoids: The number of seasonal periods in the series.
• seasonal_period: The length of the seasonal period.
• trend: In a STL decomposition of the series with $r_t$ the trend component, a quadratic model depending on time is fitted: $T_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \epsilon_t$ linearity is $\beta_1$
• curvature: In a STL decomposition of the series with $T_t$ the trend component, a quadratic model depending on time is fitted: $T_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \epsilon_t$ linearity is $\beta_2$
• e_acf1: The first autocorrelation coefficient of the remainder series in an STL decomposition of the series.
• e_acf10: The sum of the first 10 squared autocorrelation coefficients of the remainder series in an STL decomposition of the series.
• seasonal_strength: In a STL decomposition of the series with $r_t$ the remainder series and $x_t$ the detrended series: $\max[0, 1 - \text{Var}(r_t)/\text{Var}(x_t)]$
• peak: The location of the peak (maximum value) in the seasonal component of and STL decomposition of the series.
• trough: The location of the trough (minimum value) in the seasonal component of and STL decomposition of the series.
• stability: The variance of the means based on a division of the series in non-overlapping portions. The size of the portions is the frequency of the series, or 10 is the series has frequency 1.
• hw_alpha: parameter of an ets(A,A,A) model fitted on the series.
• hw_beta: parameter of an ets(A,A,A) model fitted on the series.
• hw_gamma: parameter of an ets(A,A,A) model fitted on the series.
• unitroot_kpss: The statistic for the Kwiatkowski et al. unit root test with linear trend and lag 1.
• unitroot_pp: The statistic for the “Z-alphaâ€” version of Phillips & Perron unit root test with constant trend and lag 1.

REFERENCES
[1] A. Ermagun and D. Levinson, “Spatiotemporal traffic forecasting: Review and proposed directions,” Transp. Rev., vol. 38, no. 6, pp. 786–814, 2016.
[2] J. Wang, J. Jiang, W. Jiang, C. Li, and W. X. Zhao, “LibCity: An open library for traffic prediction,” in Proc. 29th Int. Conf. Adv. Geographic Inf. Syst., Nov. 2021, pp. 145–148.
[3] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. Hoboken, NJ, USA: Wiley, 2015.

[4] M. Lippi, M. Bertini, and P. Frasconi, “Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning,” *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 871–882, Jun. 2013.

[5] H. Yao, X. Tang, H. Wei, G. Zheng, and Z. Li, “Revisiting spatio-temporal similarity: A deep learning framework for traffic prediction,” in *Proc. 33rd Conf. Artif. Intell.*, vol. 33, 2019, pp. 5668–5675.

[6] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, “Graph waveNet for deep spatio-temporal graph modeling,” in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 1907–1913.

[7] J. Bi, J. Wang, Z. Jiang, J. Ma, and H. Zhang, “Interpretable spatiotemporal deep learning model for traffic flow prediction based on potential energy fields,” in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2020, pp. 1076–1081.

[8] K. Guo, Y. Hu, Y. Sun, S. Qian, J. Gao, and B. Yin, “Hierarchical graph convolution networks for traffic forecasting,” in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 1, pp. 151–159.

[9] R. W. Liu, M. Liang, J. Nie, W. Y. B. Lim, Y. Zhang, and M. Guizani, “Deep learning-powered vessel trajectory prediction for improving smart traffic services in maritime Internet of Things,” *IEEE Trans. Netw. Sci. Eng.*, early access, Jan. 7, 2022, doi: 10.1109/TNSE.2022.3140529.

[10] R. W. Liu, M. Liang, J. Nie, Y. Yuan, Z. Xiong, H. Yu, and N. Guizani, “STMGCN: Mobile edge computing-empowered vessel trajectory prediction using spatio-temporal multi-graph convolutional network,” *IEEE Trans. Ind. Informat.*, early access, Apr. 8, 2022, doi: 10.1109/TII.2022.3165886.

[11] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–22.

[12] P. Velicković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–12.

[13] J. Zhang, Y. Zheng, and D. Qi, “Deep spatio-temporal residual networks for citywide crowd flows prediction,” in *Proc. 31st Conf. Artif. Intell. (AAAI)*, 2017, pp. 1655–1661.

[14] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,” in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–16.

[15] B. Yu, H. Yin, and Z. Zhu, “Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting,” in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3634–3640.

[16] L. Zhao, Y. Song, C. Zhang, and Y. Liu, “T-GCN: A temporal graph convolutional network for traffic prediction,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3848–3858, Sep. 2020.

[17] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, “Connecting the dots: Multivariate time series forecasting with graph neural networks,” in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 753–763.

[18] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, “Attention based spatio-temporal graph convolutional networks for traffic flow forecasting,” in *Proc. 33rd Conf. Artif. Intell. (AAAI)*, 2019, pp. 922–929.

[19] C. Zheng, X. Fan, C. Wang, and J. Qi, “GMAN: A graph multi-attention network for traffic prediction,” in *Proc. 34th Conf. Artif. Intell. (AAAI)*, 2020, vol. 34, no. 1, pp. 1234–1241.

[20] B. Lu, X. Gan, H. Jin, L. Fu, and H. Zhang, “Spatiotemporal adaptive gated graph convolution network for urban traffic flow forecasting,” in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 1025–1034.

[21] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, “Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4883–4894, Nov. 2019.

[22] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, “Adaptive graph convolutional recurrent network for traffic forecasting,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 17804–17815.

[23] S. Ravi and H. Larochelle, “Optimization as a model for few-shot learning,” in *Proc. 4th Int. Conf. Learn. Represent.*, 2016, pp. 1–6.

[24] J. Zhang, F. Chen, Y. Guo, and X. Li, “Multi-graph convolutional network for short-term passenger flow forecasting in urban rail transit,” *IET Intell. Transp. Syst.*, vol. 14, no. 10, pp. 1210–1217, Oct. 2020.

[25] R. D. Medrano and J. L. Aznarte, “On the inclusion of spatial information for spatio-temporal neural networks,” *Neural Comput. Appl.*, vol. 33, no. 21, pp. 14723–14740, 2021.

[26] M. Xu, W. Dai, C. Liu, X. Gao, W. Lin, G.-J. Qi, and H. Xiong, “Spatial-temporal transformer networks for traffic flow forecasting,” 2020, arXiv:2001.02908.

[27] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proc. 22nd Acm Sigkdd Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 785–794.

[28] J. Mockus, *Bayesian Approach to Global Optimization: Theory and Applications*, vol. 37. Berlin, Germany: Springer, 2012.