Bidirectional Spatial-Temporal Adaptive Transformer for Urban Traffic Flow Forecasting

Changlu Chen, Yanbin Liu, Ling Chen, *Senior Member, IEEE*, and Chengqi Zhang, *Senior Member, IEEE*

***Abstract*— Urban traffic forecasting is the cornerstone of the intelligent transportation system (ITS). Existing methods focus on spatial-temporal dependency modeling, while two intrinsic properties of the traffic forecasting problem are overlooked. First, the complexity of diverse forecasting tasks is nonuniformly distributed across various spaces (e.g., suburb versus downtown) and times (e.g., rush hour versus off-peak). Second, the recollection of past traffic conditions is beneficial to the prediction of future traffic conditions. Based on these properties, we propose a bidirectional spatial-temporal adaptive transformer (Bi-STAT) for accurate traffic forecasting. Bi-STAT adopts an encoder–decoder architecture, where both the encoder and the decoder maintain a spatial-adaptive transformer and a temporal-adaptive transformer structure. Inspired by the first property, each transformer is designed to dynamically process the traffic streams according to their task complexities. Specifically, we realize this by the recurrent mechanism with a novel dynamic halting module (DHM). Each transformer performs iterative computation with shared parameters until DHM emits a stopping signal. Motivated by the second property, Bi-STAT utilizes one decoder to perform the *present* → *past* recollection task and the other decoder to perform the *present* → *future* prediction task. The recollection task supplies complementary information to assist and regularize the prediction task for a better generalization. Through extensive experiments, we show the effectiveness of each module in Bi-STAT and demonstrate the superiority of Bi-STAT over the state-of-the-art baselines on four benchmark datasets. The code is available at https://github.com/chenchl19941118/Bi-STAT.git.**

***Index Terms*— Dynamic halting mechanism, spatial-temporal, transformer, urban traffic forecasting.**

I. Introduction

URBAN traffic forecasting is of great importance to the intelligent transportation system (ITS), which plays an indispensable role in the development of smart city [1]. Accurately forecasting traffic conditions will provide functional and reliable insights for intelligent urban planning and traffic management, which includes a diverse assortment of practical applications, such as mitigating traffic congestion to improve traffic efficiency, providing early warning for crowded areas to ensure public safety, and offering advice to facilitate daily commuting.

Traffic forecasting aims to predict future traffic conditions (e.g., traffic flow or traffic speed) based on the present traffic conditions recorded by a multitude of sensors spreading across the urban roads. It is a very challenging task due to the complex spatial-temporal dependency and pattern. Spatially, the traffic conditions of nearby roads have a dynamic influence on (or dependency on) each other. Temporally, the traffic conditions exhibit elusive patterns due to various factors, such as weather, rush hour, weekends, and holidays. Moreover, the spatial dependency and the temporal pattern are entangled in the traffic forecasting task of different roads at diverse time steps. Due to these challenges, existing methods focus on spatial-temporal dependency modeling. In particular, traffic forecasting is modeled as a spatial-temporal graph modeling problem [2], [3], where all sensors on the roads constitute a spatial-temporal graph to represent the evolving traffic conditions. In order to predict the future traffic condition of each node, recent studies on spatial-temporal graph modeling mainly integrate graph convolution networks (GCNs) [4] with different time-series models, i.e., recurrent neural networks (RNNs) [5], [6], convolution neural networks (CNNs) [2], [7]–[10], and transformer [11]–[15]. Both the graph structure and the present traffic condition are incorporated for future condition prediction. While existing methods show the effectiveness of modeling the spatial-temporal dependency, two intrinsic properties of the traffic forecasting problem are overlooked, which hinders existing methods from becoming more efficient and accurate traffic condition predictors.

First, the complexities of diverse forecasting tasks are imbalanced with respect to both spaces (e.g., suburb versus downtown) and times (e.g., rush hour versus off-peak). As shown in Fig. 1, the traffic flow of road 1 (in downtown) exhibits higher values and more complex patterns compared with those of road 3 (in the suburb). Meanwhile, for a specific road, the rush hour flow, oscillating frequently, is harder to predict than the off-peak flow, which shows a stable and simple pattern. As a result, the imbalanced task complexity poses considerable difficulties for accurate traffic forecasting. However, existing

Manuscript received 27 November 2021; revised 3 April 2022; accepted 12 June 2022. Date of publication 30 June 2022; date of current version 6 October 2023. This work was supported in part by ARC under Grant DP180100966. *(Corresponding author: Ling Chen.)*

Changlu Chen, Ling Chen, and Chengqi Zhang are with the Australian Artificial Intelligence Institute, University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: changlu.chen@student.uts.edu.au; ling.chen@uts.edu.au; chengqi.zhang@uts.edu.au).

Yanbin Liu is with the UWA Centre for Medical Research, The University of Western Australia, Perth, WA 6009, Australia (e-mail: csyanbin@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3183903.

Digital Object Identifier 10.1109/TNNLS.2022.3183903

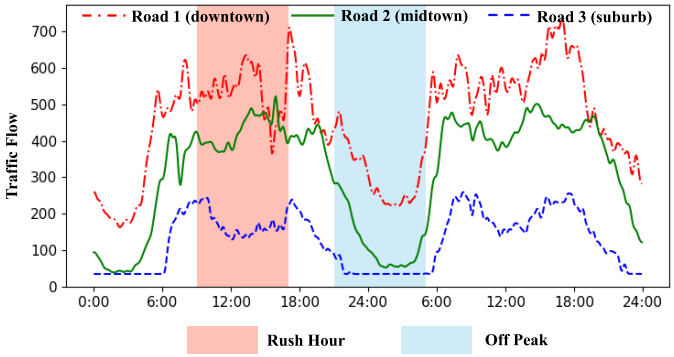


Fig. 1. Traffic forecasting tasks exhibit imbalanced complexities across varying spaces and times: 1) downtown (e.g., road 1) shows higher values and more complex patterns than suburb (e.g., road 3) and 2) the rush hour oscillates wildly, while the off-peak shows a stable and simple pattern.

methods allocate equal computation load to all tasks regardless of their spatial-temporal complexities. Consequently, important details are neglected for complex tasks due to limited computation, while excessive computation loads are wasted for simple tasks. Thus, a dynamic and adaptive computation model is required to address the aforementioned issues.

Second, the recollection of past traffic conditions plays an important role in the prediction of future traffic conditions. Recent neuroscience studies [16], [17] find that remembering the past and predicting the future are intimately linked in a common brain network. This intuition is also verified and incorporated in various practical fields, such as natural language processing (NLP) [18] and video representation [19]. However, in traffic forecasting, existing methods only focus on the future prediction task, ignoring the effect of past recollection. Since the traffic forecasting model is usually trained in one period and tested in another period, only considering the future prediction will cause the model to overfit the training period. This issue can be alleviated by the incorporation of the past recollection task, which will prevent the model from aggressively fitting the future prediction and bring extra context information for consistent prediction.

Motivated by the two properties, we propose a bidirectional spatial-temporal adaptive transformer (Bi-STAT) for accurate traffic forecasting. Specifically, Bi-STAT targets the traffic flow forecasting problem, which exhibits the dynamic varying spatial-temporal patterns of the traffic flow information shown in Fig. 1. Bi-STAT is designed to be an encoder–decoder architecture. Each encoder (decoder) maintains a spatial-adaptive transformer to model the spatial dependency and a temporal-adaptive transformer to capture the temporal pattern, which is then entangled to explore the spatial-temporal correlation. To address the first property, each transformer is devised to dynamically process the traffic streams according to the varying task complexity. Specifically, we first utilize the recurrent inductive bias of RNNs to perform iterative computation with shared parameters. Then, a novel dynamic halting module (DHM) is devised to estimate the required computation for each task and terminate the computation when necessary. The recurrent design is parameter-efficient, while the DHM is flexible and efficient by parsimoniously assigning the essential computation load for each task. To cope with the second property, at training time, we employ two independent decoders to perform the *present* $\rightarrow$ *past recollection* task and the *present* $\rightarrow$ *future prediction* task, respectively. The recollection task provides extra context information and serves as a regularizer to prevent the prediction task from excessively fitting the future stream. It is notable that we remove the DHM module in our recollection decoder. The reason is that we count on this DHM in the future prediction decoder to dynamically allocate the computation resources for better learning the future traffic representation. By contrast, we only need the past recollection decoder to serve as a simple regularizer by learning the historical traffic representation. The different mission of the two decoders leads to the difference in architecture design, which allows our model to focus more on the target future prediction task and achieve better performance gains.

The contributions of this work are summarized as follows.

1) This is the first work to dynamically tackle the traffic forecasting problem according to the unique spatial-temporal complexity, which is realized by the proposed Bi-STAT.
2) We design a novel DHM to parsimoniously assign the essential computation load for each task. DHM dynamically terminates the iterative running of the recurrent transformer, thus being flexible and parameter-efficient.
3) In contrast to existing methods, we simultaneously recollect the past traffic conditions and predict the future traffic conditions. This design is coherent with the brain mechanism and further improves traffic forecasting performance.
4) We perform comprehensive experiments on four benchmark datasets. The results show that the proposed Bi-STAT significantly outperforms the state of the art.

II. Related Work

A. Urban Traffic Forecasting

In early studies, traditional time-series prediction methods, such as historical average (HA) [20], autoregressive integrated moving average (ARIMA) [21], and VAR [22], are proposed to perform the traffic forecasting task. However, these traditional methods struggle to capture the complex nonlinear spatial-temporal correlations hidden in the large-scale traffic streams. Recently, the development of deep learning [23] provides a promising direction to solve this problem because of its capability to extract discriminative features and the ability to capture complicated spatial-temporal correlations [24]–[26]. Therefore, an assortment of deep neural networks is applied to the traffic forecasting problem, such as RNNs [27], [28], GCNs [4], [29], CNNs [23], attention, and transformers [30].

RNNs [27] have a natural superiority in modeling sequential dependencies and, thus, have been widely applied to traffic prediction problem [5], [6], [31]–[34]. In particular, the long short-term memory (LSTM) and the gated recurrent unit (GRU) are two popular RNNs variants that have been adopted to learn the temporal dependencies of the traffic

streams. For example, Zhang and Patras [32] utilize the ConvLSTM structure to model long-term trends and short-term variations of the mobile traffic volumes. ST-MetaNet [31] employs GRU as a concrete example of RNNs in their method to encode the temporal dynamics of urban traffic.

However, traffic forecasting is not a simple time-series prediction problem; it also involves a complicated spatial traffic network. The traffic network is constructed by all the sensors on the roads, which have irregular and elusive interdependencies. In order to effectively model the spatial correlations among roads, GCNs [4] have been introduced in traffic forecasting, such as [2], [5]–[7], and [35]–[40]. For example, DCRNN [5] reformulates the spatial dependency in traffic forecasting as a diffusion process on a directed graph, which is specifically achieved by the bidirectional random walks on the graph. To break the constraints of the predefined static graph, AGCRN [6] devises two adaptive modules and incorporates them to a GRU model to form a unified traffic forecasting model. Therefore, it can capture node-specific spatial and temporal correlations in the traffic streams.

CNNs have been recently applied to time-series prediction problems, such as temporal convolution networks. Compared with RNNs, CNNs can be easily combined with GCNs to jointly model the spatial and temporal patterns, which is more suitable for traffic forecasting. For example, Graph-WaveNet [2] develops an adaptive dependency matrix to complement the potential dependency that may be lost in a fixed graph matrix. They also employ the dilated causal convolution to capture long-range temporal sequences. STGCN [7] is the first study to apply convolutional structures to capture both the spatial and temporal patterns. It is designed to be a multiconvolutional block architecture, which employs the graph convolutional layers and convolutional sequence layers to model the spatial and temporal dependencies, respectively.

B. Attention and Transformer

Transformer [30] has been widely used in various spatial and temporal applications, such as NLP [18], [41], image classification [42], [43], and video representation [44]. The core building block of the transformer is the attention mechanism, which is able to dynamically learn the adaptive correlations between the input features and gather the auxiliary information from the most contributing ones for output features. Compared with RNNs, the transformer enjoys more efficient computation due to the parallelization-in-time mechanism, while RNNs require the iterative computation along with time steps. This way, the transformer can better handle the long sequence modeling than RNNs and avoid the gradient exploding and vanishing problems.

Due to the prominent sequence modeling ability, the transformer has been applied to the traffic forecasting problem recently [11]–[15], [45]. These works either design task-specific attention modules similar to the transformer or directly utilize the transformer architecture. ASTGCN [11] employs a spatial-temporal attention mechanism to learn the dynamic spatial-temporal correlations of traffic data. STTNs [12] proposes a spatial transformer to dynamically model directed spatial dependencies and a temporal transformer to capture the long-range temporal dependencies over time. In order to model the periodicity inherent in the time series, the traffic transformer [45] proposes a deep learning model to capture the continuity and periodicity in the time series, where the spatial dependencies are captured by the GCN model, and the transformer is leveraged to model the temporal dependencies.

Motivated by encoder–decoder architecture in the vanilla transformer, GeoMAN [13] proposes a multilevel attention mechanism based on an encoder–decoder architecture. The encoder consists of two different spatial attentions to model the local and global relations, while the decoder consists of temporal attention to model the dynamic temporal correlation between different time intervals. GMAN [14] proposes a graph multiattention network for traffic prediction, which follows the encoder–encoder architecture in the transformer and applies it to both the spatial and temporal dimensions to capture the dynamical dependencies.

To avoid the large accumulative errors brought by the autoregressive decoding of the canonical transformer model, NAST [15] proposes a query generation block (QGB) module between the encoder and the decoder to generate queries for the spatial-temporal transformer decoder.

Although existing methods design various spatial-temporal transformer architectures to model the spatial correlations and temporal patterns, none of them considers the dynamic task complexity with respect to both spaces (e.g., downtown versus suburb) and times (e.g., rush hour versus off-peak). In contrast, we propose a dynamic and adaptive computation model (i.e., Bi-STAT) based on the recurrent mechanism and a novel DHM. The recurrent design is parameter-efficient, while the DHM is flexible and computation-efficient. Moreover, existing methods mainly focus on the future prediction task with a transformer decoder, which incurs the risk of overfitting since the training and testing durations are always nonoverlapping. To alleviate this issue, we propose an extra recollection decoder to reconstruct the past traffic streams, which prevents the future prediction decoder from aggressively fitting the future streams and improves the generalization.

III. METHOD

A. Problem Formulation

We target at the multistep traffic flow forecasting problem, i.e., predicting the future traffic flow conditions of multiple time steps simultaneously.

Different from time-series prediction tasks, traffic forecasting is conditioned on the traffic network. Formally, a traffic network is represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$, where $\mathcal{V}$ is the set of nodes representing the traffic sensors, $|\mathcal{V}| = N$ is the number of sensors, $\mathcal{E}$ is the set of edges, and $A \in \mathbb{R}^{N \times N}$ is the adjacent matrix of the graph to measure the proximity between nodes (i.e., spatial correlations).

Based on the traffic network, we can define the traffic flow forecasting problem for all N sensors. First, different from existing traffic forecasting models that only utilize the present traffic data to predict the future traffic

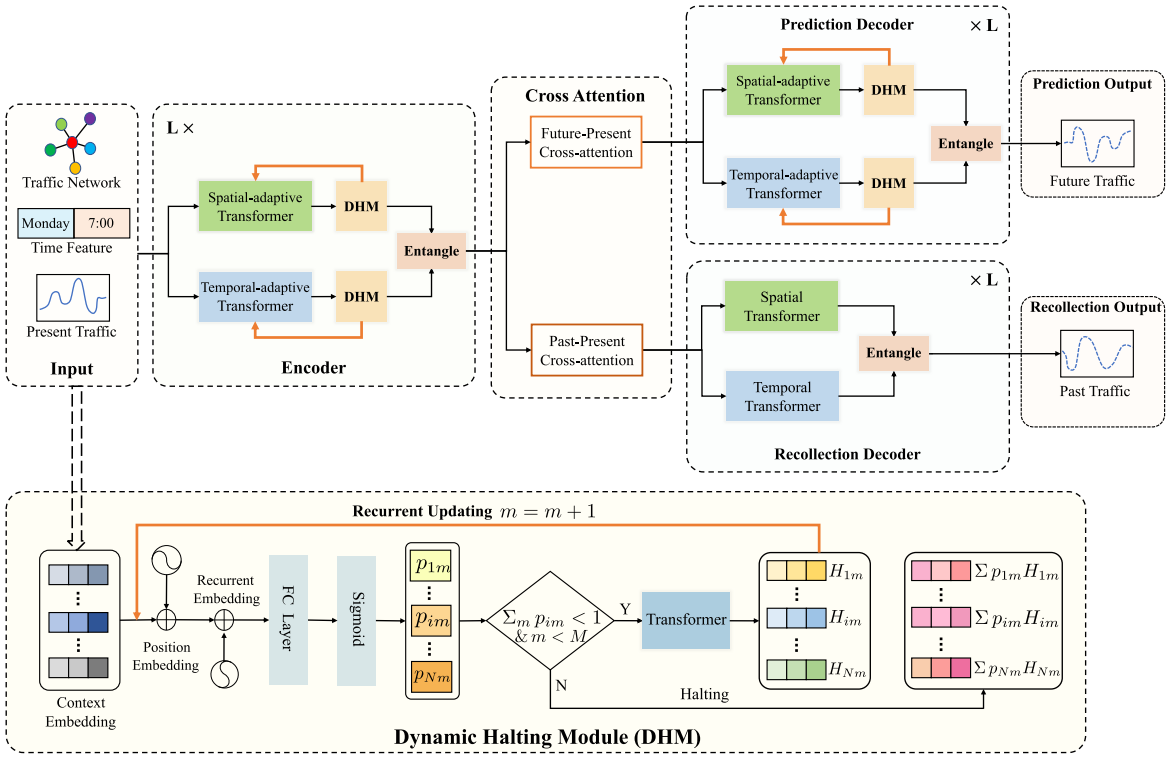


Fig. 2. Framework of the Bi-STAT. Bi-STAT consists of an encoder, a prediction decoder, and a recollection decoder. In both the encoder and prediction decoder, we realize the task-adaptive computation (spatially and temporally) by the recurrent mechanism with a novel DHM. The recollection decoder performs the past reconstruction task to provide extra context information and regularize the future prediction model.

conditions, we adopt both the historical traffic sequences $\mathcal{X}^H = (X_1, X_2, \ldots, X_H) \in \mathbb{R}^{H \times N \times d}$ and the present traffic sequences $\mathcal{X}^P = (X_{H+1}, X_{H+2}, \ldots, X_{H+P}) \in \mathbb{R}^{P \times N \times d}$. Here, H and P stand for the historical and present time steps, respectively, and d denotes the number of features in traffic conditions (e.g., traffic flow and traffic speed), which is 1 in our method to only represent traffic flow. Based on $\mathcal{X}^H$ and $\mathcal{X}^P$, our goal is to predict the conditions of future F time steps, denoted as $\mathcal{X}^F = (X_{H+P+1}, X_{H+P+2}, \ldots, X_{H+P+F}) \in \mathbb{R}^{F \times N \times d}$. To achieve this goal, we train a model f parameterized by θ to predict an approximation of $\mathcal{X}^F$ as follows:

$$\hat{\mathcal{X}}^F = f_\theta(\mathcal{X}^H, \mathcal{X}^P; \mathcal{G}). \tag{1}$$

To be consistent with existing methods, at testing time, we only utilize $\mathcal{X}^P$ and $\mathcal{G}$ for prediction. The effect of $\mathcal{X}^H$ is to regularize the model f and provide extra context information at training time.

B. Framework Overview

The overview of the Bi-STAT framework is shown in Fig. 2.

There are three sources of inputs in our traffic forecasting problem: the traffic network (***spatial context***), the time feature (***temporal context***), and the present traffic streams (***input feature***), which we will elaborate in Section III-C. After preprocessing and feature embedding, we input embeddings to the encoder. The encoder contains multiple layers with each layer maintaining a spatial-adaptive transformer, a temporal-adaptive transformer, and a DHM architecture. In the encoder, the task-adaptive computation is realized by the recurrent transformer computation with the DHM to dynamically control the number of computation steps. After that, the spatial and temporal features are entangled to capture the interdependencies among the traffic nodes at different times.

Bi-STAT contains two decoders. The prediction decoder shares the same architecture with the encoder and is used for future traffic condition prediction. The recollection decoder has a simpler structure without the DHM and serves as the regularizer to prevent the prediction decoder from aggressively fitting the future traffic conditions. Between the encoder and the decoder, we design the cross-attention module, which models the direct relationship between each future (past) time step and all the present time steps. This design disconnects the links among different future (past) time steps, which has a potential effect of mitigating the notorious error propagation issue in the traffic forecasting problem.

C. Preprocessing and Input Embedding

In this section, we describe the three kinds of inputs mentioned in III-B in detail. The utilization of the spatial and temporal context information enables our model to be aware of the spatial-temporal surroundings of the prediction task. Moreover, these contexts offer important clues for the spatial-temporal complexity of the prediction task, which can be captured by our model to dynamically assign the computation load based on task complexity.

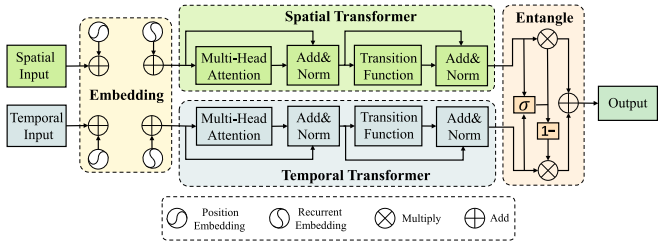


Fig. 3. STAT architecture.

1) Spatial Context: We first construct the traffic network $\mathcal{G}$. Specifically, we calculate all the pairwise Euclidean distances among all the traffic sensors in the traffic network. These distances then form the adjacent matrix A of the graph. To convert the graph nodes (i.e., traffic sensors) to feature embeddings, we adopt the widely used *node2vec* [46] approach. With *node2vec*, the graph nodes are represented by the spatial embedding (SE). Since SE is static, we feed it into a two-layer fully connected neural network to obtain the learnable spatial embeddings $\text{SE} \in \mathbb{R}^{N\times D}$.

2) Temporal Context: To build the temporal context, we employ two important temporal signals: *hour in a day* and *day in a week*. These two signals offer both short- and long-term context information for the traffic prediction task. For example, the rush hour and weekend are strong signals to influence the short- and long-term temporal evolutions. In particular, a day is divided into T_D time steps, and a week is divided into seven days. They are then encoded as one-hot vectors of dimension $\mathbb{R}^{T_D}$ and $\mathbb{R}^7$, respectively, which are concatenated as $\mathbb{R}^{T_D+7}$ and fed into a two-layer fully connected neural network. For different traffic sequences $\mathcal{X}^H$, $\mathcal{X}^P$, and $\mathcal{X}^F$, we get the corresponding temporal embedding as $\text{TE}^H \in \mathbb{R}^{H\times D}$, $\text{TE}^P \in \mathbb{R}^{P\times D}$, and $\text{TE}^F \in \mathbb{R}^{F\times D}$, respectively.

3) Input Feature: It is the original present traffic sequence $\mathcal{X}^P \in \mathbb{R}^{P\times N\times d}$. We also utilize a two-layer fully connected neural network to align the dimension with the spatial and temporal embeddings SE and TE^P. After that, we get $\mathcal{X}_1^P \in \mathbb{R}^{P\times N\times D}$.

With all the SEs, temporal embedding TE^P, and input feature $\mathcal{X}_1^P$ at hand, we can obtain the final input embeddings $E \in \mathbb{R}^{P\times N\times 2D}$ as follows:

$$E[i,j,:] = \text{Concat}\big(\mathcal{X}_1^P[i,j,:], \text{SE}[j,:] + \text{TE}^P[i,:]\big). \quad (2)$$

D. Spatial-Temporal Adaptive Transformer

To effectively capture the spatial-temporal correlation and patterns from the traffic sequences, we propose an STAT, as shown in Fig. 3. Specifically, we devise a spatial transformer and a temporal transformer to process the spatial input and temporal input, respectively. Before the spatial (temporal) inputs are fed into the spatial (temporal) transformer, the position embedding and recurrent embedding are incorporated to provide the context information (e.g., sensor identity, time step, and recurrent step). Specifically, the transformer is composed of the multihead attention, layer normalization with residual design, and the transition function. After the spatial transformer and temporal transformer processing, we further design an entangle module to model the intercorrelations between the spatial and temporal sequences.

Before describing the transformer details, we first discuss the design choice for dynamic computation. A straightforward design for dynamic computation is to first expand multiple sequential transformer layers and then dynamically control the number of layers according to the task complexity. However, this design has two disadvantages. First, directly expanding the transformer layers will linearly increase the parameter number. If a large computation load is required, the transformer will be inefficient in terms of parameter number and become unaffordable. Second, with dynamic computation, the deeper layers will receive less training data than the shallower layers, being less effective. To address these issues, we propose a different design in this article. Specifically, we utilize the *recurrent inductive bias* of the RNNs, which performs recurrent transformer updating with shared parameters. For dynamic computation, we devise a DHM to control the number of recurrent updating steps (which will be described in Section III-E). This design enjoys the efficiency of the recurrent parameter-sharing mechanism and the flexibility of the dynamic computation by the DHM.

In the following, we first describe how to adapt the existing position embedding of the transformer to handle the recurrent transformer updating. Then, we describe the architecture of the spatial transformer, the temporal transformer, and the entangle module, respectively.

1) Position Embedding and Recurrent Embedding: In the original design of transformer, position embedding is devised to contain the relative position information of the sequence. Formally, the position embedding is computed as follows:

$$\text{PE}(p, 2i) = \sin\big(p/10\,000^{2i/D}\big) \quad (3)$$

$$\text{PE}(p, 2i+1) = \cos\big(p/10\,000^{2i/D}\big) \quad (4)$$

where p is the position index and i is the dimension of the embedding. In order to incorporate the recurrent step information into the transformer, we devise a 2-D (position, step) coordinate embedding. For the positions $1 \le p \le T$ and recurrent steps $1 \le m \le M$, the coordinate embedding is computed as follows:

$$\text{PE}^m_{p,2i} = \sin\big(p/10\,000^{2i/D}\big) + \sin\big(m/10\,000^{2i/D}\big) \quad (5)$$

$$\text{PE}^m_{p,2i+1} = \cos\big(p/10\,000^{2i/D}\big) + \cos\big(m/10\,000^{2i/D}\big). \quad (6)$$

In (5) and (6), the first sin / cos calculates the position embedding, while the second sin / cos computes the recurrent embedding indicating the recurrent computation step.

2) Temporal-Adaptive Transformer: Attention is the core component of transformer. In this article, we use the scaled dot-product attention. Formally, the attention function combines queries Q, keys K, and values V as follows:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{D}}\right)V \quad (7)$$

where D is the number of columns of Q, K, and V. A widely used variant of attention in transformer is the multihead

attention with k heads, i.e.,

$$\begin{aligned}&\text{MultiHeadAttention}\big(H^m\big)\\&\quad=\text{Concat}(\text{head}_1,\ldots,\text{head}_K)W^O \quad (8)\\&\text{where head}_i=\text{Attention}\Big(H^mW_i^Q,H^mW_i^K,H^mW_i^V\Big). \quad (9)\end{aligned}$$

Here, H^m stands for current state features with m denoting the recurrent computation step, $W_i^Q\in\mathbb{R}^{D\times D/k}$, $W_i^K\in\mathbb{R}^{D\times D/k}$, and $W_i^V\in\mathbb{R}^{D\times D/k}$.[1]

To include the recurrent inductive bias of RNNs into the transformer, we perform recurrent updating for the transformer state H^m. Specifically, at the recurrent step m, we update the state H^m as follows:

$$\begin{aligned}H^m&=\text{LayerNorm}\big(A^m+\text{Transition}\big(A^m\big)\big) \quad (10)\\\text{where } A^m&=\text{LayerNorm}\big(\big(H^{m-1}+\text{PE}^m\big)\\&\quad+\text{MultiHeadAttention}\big(H^{m-1}+\text{PE}^m\big)\big) \quad (11)\end{aligned}$$

where LayerNorm(·) [47] is the layer normalization adopted to accelerate the training of model and Transition(·) is the transition function that consists of two affine transformations with an ReLU activation between them.

3) Spatial-Adaptive Transformer: The spatial-adaptive transformer owns a similar structure as the temporal-adaptive transformer but has two differences in order to capture the spatial dependencies among traffic sensors.

First, the position embedding is different. In the temporal-adaptive transformer, the position embedding represents the time steps of the traffic sequences so that p ranges from 1 to T, $1\le p\le T$ (e.g., $T=P$ for the present sequence $\mathcal{X}^P$). For the spatial-adaptive transformer, we utilize position embedding to encode the sensor identities, i.e., $1\le p\le N$.

Second, the effective self-attention dimension in the transformer is different. For the temporal-adaptive transformer, we reshape $\mathcal{X}^P$ to $\mathbb{R}^{N\times P\times D}$, where N is the batch dimension and P is the self-attention dimension for the temporal correlation, while, for the spatial-adaptive transformer, $\mathcal{X}^P$ is reshaped to $\mathbb{R}^{P\times N\times D}$, where P is the batch dimension and N is the self-attention dimension for the spatial correlation.

4) Entangle Module: The spatial-adaptive transformer and the temporal-adaptive transformer, respectively, capture the spatial dependencies and temporal patterns in their own models. However, due to the complexity of the traffic forecasting problem, the spatial dependencies and the temporal patterns are usually entangled to exhibit complicated spatial-temporal correlations. For example, the rush hour of a downtown road will show the common patterns of both the downtown area spatially and the rush hour temporally.

Let H_S and H_T denote the features after the spatial transformer and the temporal transformer, respectively. We first utilize a linear model with sigmoid activation to obtain the entangle gate g as follows:

$$g=\sigma\big(H_S\mathbf{W}_S+H_T\mathbf{W}_T+\mathbf{b}_g\big) \quad (12)$$

where $\mathbf{W}_S\in\mathbb{R}^{D\times D}$, $\mathbf{W}_T\in\mathbb{R}^{D\times D}$, and $\mathbf{b}\in\mathbb{R}^D$ are learnable parameters, and $\sigma(\cdot)$ is the sigmoid activation. Then, the spatial features and temporal features are combined as

$$H=g\odot H_S+(1-g)\odot H_T \quad (13)$$

where $\odot$ denotes the elementwise product.

E. Dynamic Halting Module

As described above, we utilize the recurrent updating for transformer layers to realize the parameter-sharing recurrent computation. The recurrent mechanism increases the computation load to solve difficult and complex prediction tasks, while it also keeps the parameter number unchanged, thus being parameter-efficient. However, the disadvantage is that the large computation load will be a waste for a simple prediction task. To solve this issue, we propose a DHM to parsimoniously assign the necessary computation load according to the task complexity.

To be aware of the spatial-temporal task complexity, the initial input of DHM requires to contain the spatial context, the temporal context, and the input feature, which coincides with the input embeddings E obtained in Section III-C. Moreover, since the DHM is applied to control the recurrent computation of the transformer, we should also include the position embedding and recurrent embedding to inform the DHM the current status of the recurrent computation. Through this analysis, we find that the transformer state at a specific recurrent step, H^m, contains all the aforementioned context information to evaluate the spatial-temporal task complexity. Thus, we directly utilize H^m for the DHM.

Note that the spatial and temporal transformers own different DHMs. Here, we take the DHM attached to temporal transformer as an example. Specifically, at each recurrent step m, given H^m, we first calculate the halting probability p^m as follows:

$$p^m=\sigma\big(H^m\mathbf{W}_h+b_h\big) \quad (14)$$

where $\mathbf{W}_h\in\mathbb{R}^D$, $b_h\in\mathbb{R}^1$, and $p^m\in\mathbb{R}^P$ indicates the halting unit for all the P present sequences. Let p_i^m stand for the ith sequence in particular. To determine if the computation step for the ith sequence should be stopped to save the resource, we accumulate the halting probability over the m' recurrent steps as $\sum_{m=1}^{m'}p_i^m$. The stopping criteria are defined as follows:

$$\begin{cases}\text{stop for sequence } i, & \text{if } m=M \text{ or } \displaystyle\sum_{m=1}^{m'}p_i^m>=1-\epsilon\\ \text{continue}, & \text{otherwise}\end{cases} \quad (15)$$

where M is the maximum number of recurrent steps used to rigidly limit the computation costs and ϵ is a small constant to allow halting after a single recurrent step if $p_i^1>=1-\epsilon$.

With (15), we can define the real computation steps for each sequence i and the remainder value $R(i)$ as follows:

$$N(i)=\min\left\{M,\min\left\{m':\sum_{m=1}^{m'}p_i^m\ge 1-\epsilon\right\}\right\} \quad (16)$$

$$R(i)=1-\sum_{i=1}^{N(i)-1}p_i^m. \quad (17)$$

[1]The input layer weight dimensions are $\mathbb{R}^{2D\times D/k}$ since $E\in\mathbb{R}^{P\times N\times 2D}$.

From (16), we see that the expected recurrent steps will be probably smaller than M.

To make full use of the halting probability, we utilize the weighted state features to represent H^m

$$H_T[i] = \sum_{i=1}^{N(i)} p_i^m H^m[i]. \tag{18}$$

According to this weighted state representation, the halting probability not only controls the dynamic computation but also reweights the feature learning procedure.

F. Cross Attention Module

In sequence prediction tasks, a common strategy is to chronologically predict the sequence step by step and employ the prediction result from the previous step as the input of prediction for the next step. This manner, however, will cause the notorious error propagation effect between different prediction steps in the long-term prediction, which has been observed in previous traffic forecasting work [6], [12], [14], [30]. To address this issue, we propose a future-present cross-attention module between the encoder and the prediction decoder, as well as a past-present cross-attention module between the encoder and the recollection decoder, respectively. The cross attention module directly models the relationship between each future (past) time step and all the present time steps. Due to space constraint, we only take the future-present module as an example, and the formulation for the past-present module is similar. Formally, given the present time steps P_i ($P_i = 1, \ldots, P$), the future time steps F_i ($F_i = 1, \ldots, F$), and a sensor s_i, the spatial-temporal contexts for step P_i and step F_i are $\mathrm{SE}[s_i, :] + \mathrm{TE}^P[P_i, :]$ and $\mathrm{SE}[s_i, :] + \mathrm{TE}^F[F_i, :]$, respectively. We then model the correlation between future step F_i and present step P_i as

$$\begin{aligned} E_P &= \mathrm{SE}[s_i, :] + \mathrm{TE}^P[P_i, :] \\ E_F &= \mathrm{SE}[s_i, :] + \mathrm{TE}^F[F_i, :] \\ \lambda^{s_i}_{F_i,P_i} &= \frac{\langle E_F, E_P \rangle}{\sqrt{D}} \\ \lambda^{s_i}_{F_i,P_i} &= \frac{\exp\left(\lambda^{s_i}_{F_i,P_i}\right)}{\sum_{P_j=1}^{P} \exp\left(\lambda^{s_i}_{F_j,P_i}\right)} \end{aligned} \tag{19}$$

where f is a fully connected layer followed by the ReLU activation. With the correlation, we can recalculate the input to the decoder $H_D \in \mathbb{R}^{F \times N \times D}$ as

$$H_D[F_i, s_i, :] = \sum_{P_i=1}^{P} \lambda^{s_i}_{F_i,P_i} H[P_i, s_i, :]. \tag{20}$$

G. Decoders and Loss Function

1) Prediction Decoder: As shown in Fig. 2, the prediction decoder shares the same structure as the encoder, with details described in Section III-D. The input to the prediction decoder is the encoded future representation from the future-present cross-attention module, where the correlations among the future time steps are isolated, and each future time step independently correlates to all the present time steps. This way, the error propagation issue is alleviated.

2) Recollection Decoder: Previous traffic forecasting methods only employ the prediction decoder at training time. Due to the special property of the traffic forecasting problem, we can only utilize data from the previous and present duration for model training and receive data from the nonoverlapping future duration for testing. This training and testing inconsistency makes the prediction decoder easily overfit the training duration and hard to generalize. On the other hand, according to recent neuroscience studies [16], [17], remembering the past plays an important role in the prediction of the future.

According to the above analysis, we propose a recollection decoder to endue our model with the ability to simultaneously predict the future and reconstruct the past. For the architecture design, a straightforward design is to directly copy the prediction decoder. However, this design has two potential issues. First, while the aim of the recollection decoder is to alleviate the overfitting issue of the prediction decoder, directly adopting the design of the complex predictor decoder will increase the overall model parameters and further increase the overfitting. Second, if the recollection decoder is as powerful as the prediction decoder, then the model will try to perform the prediction and reconstruction task with the same effort. This potential competition will interfere with the future prediction task. Taking these into consideration, we devise the recollection decoder to be a simple structure: spatial transformer and temporal transformer without the DHM.

3) Loss Function: At the training time, the overall losses include the historical reconstruction loss $\mathcal{L}_H$, the future prediction loss $\mathcal{L}_F$, and the penalty loss $\mathcal{L}_{\mathrm{DHM}}$ from the DHM.

For $\mathcal{L}_H$ and $\mathcal{L}_F$, we utilize the mean absolute error (MAE) between the ground truth and the prediction (reconstruction) as follows:

$$\mathcal{L}_H = \frac{1}{H} \sum_{t=1}^{H} \left| \mathcal{X}^H - \hat{\mathcal{X}}^H \right| \tag{21}$$

$$\mathcal{L}_F = \frac{1}{F} \sum_{t=1}^{F} \left| \mathcal{X}^F - \hat{\mathcal{X}}^F \right| \tag{22}$$

where $\hat{\mathcal{X}}^H$ and $\hat{\mathcal{X}}^F$ denote the estimated reconstruction and the prediction made by our model, respectively.

For $\mathcal{L}_{\mathrm{DHM}}$, it is utilized to balance the required computation load and the model efficiency

$$\mathcal{L}_{\mathrm{DHM}} = N(i) + R(i). \tag{23}$$

Since $N(i)$ is an integer value ranging from 1 to M, it is not differentiable. In practice, we treat $N(i)$ as constant and minimize $R(i)$ instead.

Finally, we can combine all the losses as

$$\mathcal{L} = \mathcal{L}_F + \alpha \mathcal{L}_H + \beta \mathcal{L}_{\mathrm{DHM}} \tag{24}$$

where α and β are two regularization terms to control the effect of recollection decoder and the DHM.

IV. Experiments

A. Datasets and Evaluation Metrics

1) Dataset Details: To demonstrate the effectiveness of our method, we conduct experiments on four large-scale real-world

TABLE I
DATASET STATISTICS

Dataset	# of sensors	Time duration
PeMSD3	358	09/01/2018 - 11/30/2018
PeMSD4	307	01/01/2018 - 02/28/2018
PeMSD7	883	05/01/2017 - 08/31/2017
PeMSD8	170	07/01/2016 - 08/31/2016

traffic forecasting datasets: PeMSD3, PeMSD4, PeMSD7, and PeMSD8. These datasets were collected by The California Department of Transportation (Caltrans) through its freeway performance measure system (PeMS) [48]. The detailed statistics of the four datasets are shown in Table I. Due to their scale, diversity, and rich traffic conditions, these datasets are the most widely used benchmarks for traffic forecasting problem [6], [7], [11], [36], [38], [39].

2) Preprocessing and Data Splitting: Following recent methods [2], [6], [14], [39], all the four datasets are aggregated into 5-min windows to generate 12 data points (time steps) per hour and consequently 288 data points per day. Z-score normalization is applied to the traffic data for more stable training. To make fair comparisons with existing methods, we chronologically allocate 70% of the data for the training set, 10% for the validation set, and 20% for the testing set. Note that the training, validation, and testing splits have no overlap in time duration.

3) Evaluation Metrics: We employ three commonly used metrics [6], [14], [39]: MAE, root mean square error (RMSE), and mean absolute percentage error (MAPE) to measure the performance of the prediction models.

B. Baselines

We compare the proposed Bi-STAT with the following baselines:

1) DCRNN [5], which incorporates the diffusion convolution into the GRU module for multistep traffic forecasting;
2) STGCN [7], which employs the graph convolutional layers and convolutional sequence layers;
3) ASTGCN [11], which employs the attention mechanism to capture the spatiotemporal dynamic correlations;
4) GraphWaveNet [2], which is constructed by the graph convolution layer (GCN) and the gated temporal convolution layer (gated TCN);
5) STSGCN [36], which captures the localized spatial and temporal correlations individually;
6) STFGCN [38], which constructs spatial and temporal graphs and combines them with a spatial-temporal fusion model;
7) AGCRN [6], which designs two adaptive modules and incorporates them into a unified GRU model;
8) Z-GCNET [39], which is a time-aware zigzag persistence-based graph convolutional network;
9) GMAN [14], which adopts an encoder–decoder structure to include multiple spatial-temporal attention blocks;
10) STTN [12], which incorporates spatial-temporal dependencies into the block of spatial-temporal transformer to achieve more accurate traffic forecasting;
11) traffic transformer [45], which designs an encode-decoder architecture to consistently model the spatial and temporal dependencies in the traffic time-series by GCN and transformer, respectively.

C. Experimental Setups

As described in Section IV-A, we preprocess the datasets to generate 12 data points (time steps) per hour. At training time, we utilize the historical 1-h and present 1-h traffic sequences as input and predict the future 1-h traffic sequences. To be consistent with previous methods, at testing time, only the present 1-h sequences are utilized for prediction. Specifically, we set all time steps to 12, i.e., $H = 12$, $P = 12$, and $F = 12$.

In our framework, the recollection decoder is mainly used for regularization and providing auxiliary information, so its structure is relatively simple compared to the encoder and the prediction decoder. In particular, the structure parameters involve the number of transformer layers L, the number of heads K in multihead attention, and the transformer dimension D. For the recollection decoder, we set $L = 1, K = 3$, and $D = 8$ on all the four datasets. For the encoder and the prediction decoder, we set $L = 2, K = 3$, and $D = 8$ on PeMSD4 and PeMSD8, and $L = 1, K = 3$, and $D = 8$ on PeMSD3 and PeMSD7. Moreover, we set a small loss weight α for the recollection decoder loss, i.e., $\alpha = 0.001$ for PeMSD3 and PeMSD8, and $\alpha = 0.01$ for PeMSD4 and PeMSD7.

For the DHM, we have several parameters to adaptively adjust the computation steps, i.e., maximum recurrent steps M, the stopping threshold ϵ, and the penalty weight β for DHM loss. We empirically set these parameters according to the validation set to balance the computation load and the accuracy. In particular, $M = 6$ in all experiments (unless stated otherwise), $\epsilon = 0.001$ for PeMSD3 and 0.0001 for other three datasets, and $\beta = 0.001$ for all datasets. Note that our method does not rely on exhaustively tuning these parameters. Instead, they are used to obtain a desired computation-accuracy balance. We will demonstrate this in the following.

Our model is trained by the Adam optimizer with a learning rate of 0.001 for 100 epochs. The best model checkpoint is selected by the performance of the validation set.

D. Comparison With the State-of-the-Art Methods

For a specific time step, the prediction can be generated from any previous 1-h training sequences with an interval (a.k.a. horizon) ranging from 1 to 12 steps. The small interval (horizon) is nearer to the training sequences, thus leading to a more accurate prediction. To be consistent with other methods, we average the performance of all 12 intervals (horizons).

The overall comparison with nine state-of-the-art methods is shown in Table II. It is obvious that the proposed Bi-STAT outperforms all other baselines with a large margin on all four datasets and three metrics. Among all the

TABLE II

COMPARISON WITH THE STATE-OF-THE-ART METHODS ON FOUR DATASETS REGARDING THREE METRICS (THE SMALLER THE BETTER)

Model	Backbone	PeMSD3			PeMSD4			PeMSD7			PeMSD8		
		MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)
STGCN [7]	GCN	17.49	30.12	17.15	22.70	35.55	14.59	25.38	38.78	11.08	18.02	27.83	11.40
ASTGCN [11]	GCN	17.69	29.66	19.40	22.93	35.22	16.56	28.05	42.57	13.92	18.61	28.16	13.08
STSGCN [36]	GCN	17.48	29.21	16.78	21.19	33.65	13.90	24.26	39.03	10.21	17.13	26.80	10.96
STFGNN [38]	GCN	16.77	28.34	16.30	19.83	31.88	13.02	22.07	35.80	9.21	16.64	26.22	10.60
Z-GCNETs [39]	GCN	-	-	-	19.50	31.61	12.78	-	-	-	15.76	25.11	10.01
GraphWaveNet [2]	GCN	19.85	32.94	19.31	25.45	39.7	17.29	26.85	42.78	12.12	19.13	31.05	12.68
DCRNN [5]	RNN	18.18	30.31	18.91	24.70	38.12	17.12	25.30	38.58	11.66	17.86	27.83	11.45
AGCRN [6]	RNN	16.75	28.60	16.23	19.83	32.26	12.97	21.10	34.99	8.93	15.95	25.22	10.09
GMAN [14]	Transformer	16.49	26.48	17.13	19.36	31.06	13.55	21.48	34.55	9.01	14.51	23.68	9.45
STTN [12]	Transformer	16.11	27.87	16.19	19.32	30.79	13.15	21.05	33.77	8.94	15.28	24.25	9.98
Traffic-Transformer [45]	Transformer	16.39	27.87	15.84	19.16	30.57	13.70	23.90	36.85	10.90	15.37	24.21	10.09
Bi-STAT (Ours)	Transformer	**15.30**	**25.80**	**15.46**	**18.53**	**29.96**	**12.37**	**20.28**	**33.24**	**8.50**	**13.58**	**23.08**	**9.21**

TABLE III

EFFECTIVENESS OF THE PROPOSED DHM AND THE RECOLLECTION DECODER REGARDING VARIOUS HORIZONS. HORIZON STANDS FOR THE INTERVAL BETWEEN TRAINING AND TESTING SEQUENCES, E.G., HORIZON = 3 MEANS 15 MIN

Model	Horizon=3 (15min)			Horizon=6 (30min)			Horizon=12 (60min)			Average		
	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)
Transformer	18.44	29.67	12.84	19.22	30.94	13.40	21.08	33.54	14.89	19.36	31.06	13.55
Transformer+Recollect	18.19	29.34	12.31	18.89	30.51	12.76	20.40	32.71	13.87	18.99	30.58	12.93
Transformer+DHM	17.87	28.96	12.19	18.55	30.08	12.67	19.91	31.95	13.89	18.62	30.10	12.78
Bi-STAT (Ours)	**17.81**	**28.82**	**11.80**	**18.46**	**29.93**	**12.30**	**19.79**	**31.83**	**13.39**	**18.53**	**29.96**	**12.37**

TABLE IV

PERFORMANCE AND EFFICIENCY COMPARISON BETWEEN DHM AND THE FIXED RECURRENT COMPUTATION

# Layers	Model	MAE	RMSE	MAPE (%)	$Steps_{\mathbf{S}}$	$Steps_{\mathbf{T}}$
$L = 1$	*Transformer*	19.48	31.01	13.48	1.0	1.0
	Transformer+Recurrent	19.17	30.77	12.99	6.0	6.0
	Bi-STAT (Ours)	**18.62**	**30.13**	**12.82**	1.1	4.1
$L = 2$	*Transformer*	19.36	31.06	13.55	1.0	1.0
	Transformer+Recurrent	19.04	30.64	12.78	6.0	6.0
	Bi-STAT (Ours)	**18.53**	**29.96**	**12.37**	2.4	2.8

baselines, AGCRN [6] and GMAN [14] perform relatively well. To be specific, AGCRN designs two adaptive modules to learn the dynamic node-specific correlations and utilizes RNNs for sequence prediction, while GMAN employs several spatial-temporal attention blocks similar to the transformer. Our Bi-STAT beats both AGCRN and GMAN with a large margin, corroborating the effectiveness against both RNNs-based and transformer-based strong competitors.

E. Ablation Study

1) Effectiveness of the Designed Model Components: We first investigate the effectiveness of the proposed DHM and the recollection decoder. To begin with, a simple baseline is constructed by removing both the DHM and the recollection decoder from our Bi-STAT model. We call this baseline model "***Transformer***" since only naive transformers are utilized here. Then, we add the DHM into the baseline model and call it "***Transformer*** + ***DHM***." Another variant model is the baseline model combined with the recollection decoder, called "***Transformer*** + ***Recollect***." Our model "**Bi-STAT**" incorporates all the proposed components.

We conduct experiments on the PeMSD4 dataset, and the results are shown in Table III. We can see that, after introducing the recollection decoder (i.e., *Transformer+Recollect*), the performance is improved for all horizons and the average, showing the effectiveness of the recollection decoder. This also verifies the regularization effect of the recollection decoder to encourage the prediction model for better generalization. Besides, incorporating the DHM into the transformer (i.e., *Transformer* + *DHM*) leads to large performance gains for all horizons and the average. The result demonstrates the effectiveness of DHM, which assigns adaptive computation loads according to task complexities. Our complete model, combining both the recollection decoder and DHM, further improves the performance against every single model, showing the complementary effect of the two devised components.

2) In-Depth Analysis of the Dynamic Halting Module: We further perform an in-depth analysis of the DHM. First, we study whether the performance improvements come from the increased computation or from the adaptive computation mechanism. To verify this, we construct a baseline with fixed recurrent computation steps, called "***Transformer*** + ***Recurrent***." In contrast, our model "*Bi-STAT*" adopts the DHM mechanism to adaptively stop the computation, resulting in fewer computation steps. Here, we calculate the average computation steps for the spatial transformer and the temporal transformer, denoted by $Steps_{\mathbf{S}}$ and $Steps_{\mathbf{T}}$. Also, we take the number of transformer layers into account, i.e., $L = 1$ and $L = 2$. The results of the PeMSD4 dataset are shown in Table IV. With six computation steps, *Transformer* + *Recurrent* only slightly increases

TABLE V
EFFECT OF THE DHM INSERTED INTO DIFFERENT POSITIONS OF OUR MODEL, E.G., SPATIAL TRANSFORMER OF THE ENCODER ($\text{Encoder}_{\mathbf{S}}$)

# Layers	$Encoder_{\mathbf{S}}$	$Encoder_{\mathbf{T}}$	$Decoder$	MAE	RMSE	MAPE (%)	$Steps_{\mathbf{S}}$	$Steps_{\mathbf{T}}$
$L=1$				19.48	31.01	13.48	1.0	1.0
	✓			19.07	30.65	13.06	1.6	1.0
		✓		19.17	30.78	12.88	1.0	2.0
	✓	✓		18.97	30.44	**12.74**	1.6	2.1
	✓	✓	✓	**18.76**	**30.16**	12.91	1.2	3.7
$L=2$				19.36	31.06	13.55	1.0	1.0
	✓			18.92	30.56	12.71	1.6	1.0
		✓		18.74	30.25	12.55	1.0	2.1
	✓	✓		18.70	30.15	**12.40**	1.2	4.0
	✓	✓	✓	**18.62**	**30.10**	12.78	2.5	2.8

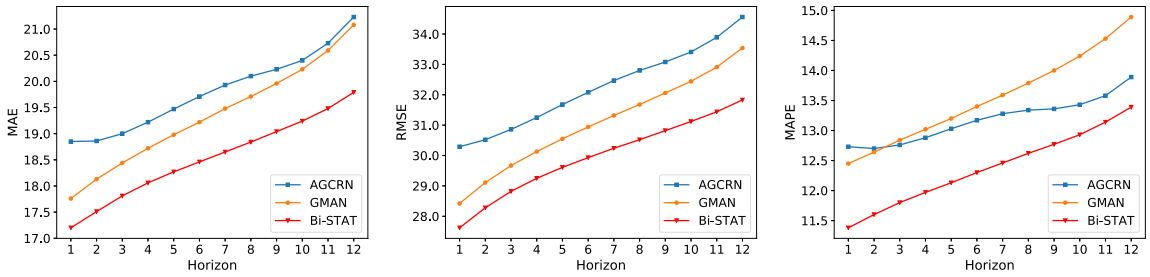


Fig. 4. Performance comparison among different methods with varying horizons on the PeMSD4 dataset w.r.t. three metrics: (a) MAE, (b) RMSE, and (c) MAPE.

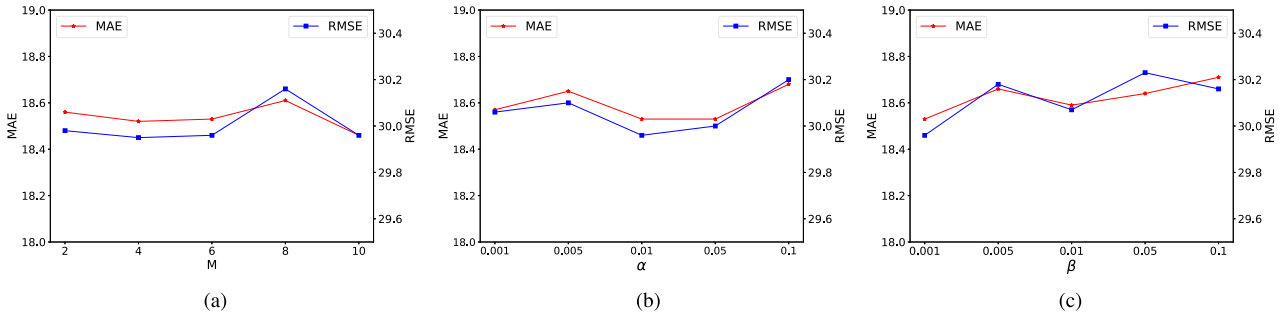


Fig. 5. Influence of three parameters w.r.t. two metrics: (a) max recurrent steps, (b) loss weight for the recollection decoder, and (c) penalty weight for the DHM.

the performance. Our *Bi-STAT* employs fewer computation steps both spatially and temporally but achieves much better performance. This demonstrates that our method does not rely on increased computation to boost performance. Instead, the adaptive computation mechanism is the key to our success.

From Table IV, we also observe that the average spatial steps and temporal steps are different. Thus, we insert DHM into different positions of our model, i.e., the spatial transformer of the encoder ($\text{Encoder}_{\mathbf{S}}$), the temporal transformer of the encoder ($\text{Encoder}_{\mathbf{T}}$), and the prediction decoder (Decoder). The results on the PeMSD4 dataset are shown in Table V. We can find that both the spatial and temporal transformers of the encoder are benefited from the DHM, verifying the effectiveness of our model in coping with dynamic spatial-temporal task complexities. The DHM also benefits the decoder, which means that the adaptive computation is also important for accurate sequence reasoning. As to computation steps, our best model only employs 2.6 steps by average, while the performance is much better than the naive transformer.

3) Parameter Analysis: To further investigate the robustness and effectiveness of our Bi-STAT method, we analyze the performance in terms of several related parameters.

Prediction Performance w.r.t. Varying Horizons: We compare two competitive methods, GMAN and AGCRN, by varying the horizon from 1 to 12, and show the results in Fig. 4. It can be observed that our method always outperforms the other two methods with a large margin. This demonstrates that our method can generate accurate predictions for both the short-term streams (small horizon) and long-term streams (big horizon).

Robustness to Hyperparameters: We study the influence of three parameters, i.e., max recurrent steps M, loss weight α for the recollection decoder, and penalty weight β for the DHM. M and β control the maximum computation load available to assign to different tasks. α controls the regularization strength. From Fig. 5, we can observe that our method is stable to the parameter changes. For each parameter, we choose the one

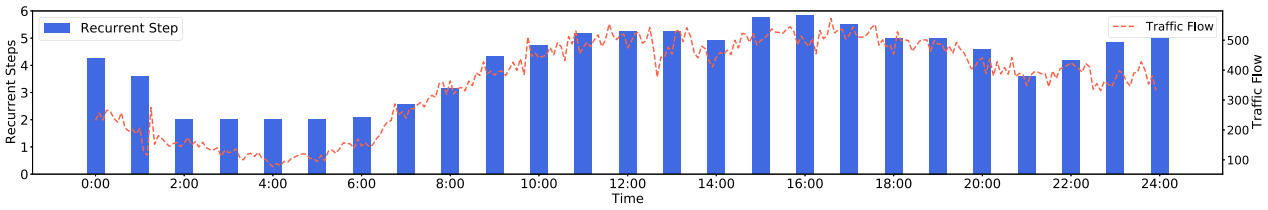
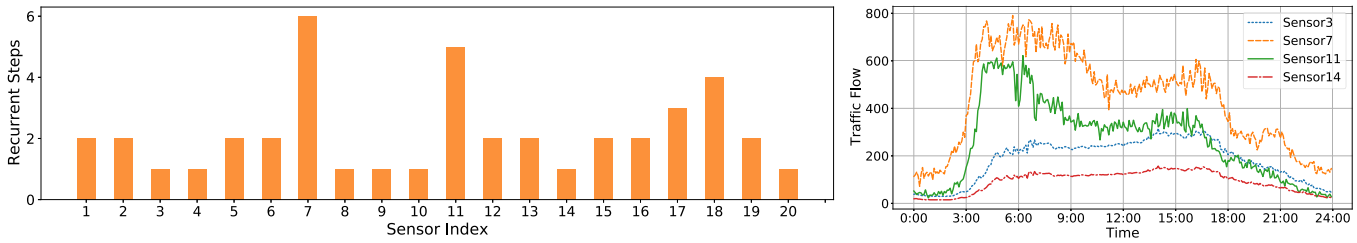


Fig. 6. Dynamic recurrent computation steps in different times (hours) of a day and the corresponding ground-truth flow.



(a) (b)

Fig. 7. Different numbers of (a) computation steps for different sensors at the same time step and (b) traffic flow truth for different sensors.

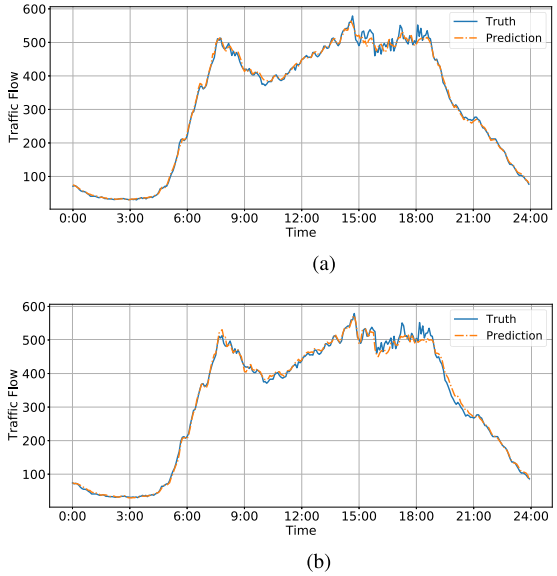
that best reflects the performance-computation balance, e.g., $M = 6, \alpha = 0.01$, and $\beta = 0.001$.

F. Visualization Results

In this section, we show some visualization results to provide a deep understanding of the mechanism of our method.

1) Dynamic Computation w.r.t. Different Spaces and Times: We first record all the recurrent computation steps of the spatial-adaptive transformer and the temporal-adaptive transformer during the testing process on the PeMSD8 dataset. Then, we postprocess these recurrent steps for better visualization. Temporally, we aggregate the recurrent steps of the temporal transformer by the hour for 24 h on a specific day (i.e., August 20, 2016). The recurrent steps are averaged for each hour and shown in the bar plot in Fig. 6. Meantime, the corresponding ground-truth flow is shown in the curve plot of Fig. 6. It can be seen that the dynamic recurrent steps assigned to each hour maintain a high degree of consistency with the patterns of the ground-truth flow (e.g., rush hour versus off-peak). The rush hours with more complex traffic patterns are assigned with more recurrent steps, while the off-peaks with simple patterns are allocated with fewer recurrent steps. This indicates that our DHM indeed learns an effective model to adaptively assign the computation loads w.r.t. the temporal task complexities.

Similarly, we also aggregate the recurrent steps of the spatial transformer corresponding to all sensors and the recurrent steps of some iconic sensors are shown in Fig. 7(a). It can be observed that most sensors have relatively small recurrent steps (e.g., sensors 3 and 14). However, there are also several sensors owing large recurrent steps (e.g., sensors 7 and 11). To further study the differences, we also plot the traffic flow of four representative sensors. Form Fig. 7(b), we can observe that the oscillating patterns of sensor 7 and sensor 11 are much more complex than those of sensor 3 and sensor 14. In this sense, it is reasonable for sensor 7 and sensor 11 to obtain large computation loads. This visualization verifies the effectiveness of our model to assign dynamic spatial computation loads for different sensors.



(a) (b)

Fig. 8. Predicted traffic flow visualization on PeMSD4 for different prediction intervals (a.k.a. horizon): (a) 5 and (b) 60 min.

2) Visualization for Both Short- and Long-Term Predictions: We show the visualization results of the predicted traffic flow on the PeMSD4 dataset for different prediction intervals in Fig. 8. For both the short-term prediction (5 min) and the long-term prediction (60 min), our method can accurately generate the prediction sequences. The visualization is consistent with our quantitative results in Table II.

G. Computation Time

Finally, we compare the computation time of Bi-STAT with AGCRN, DCRNN, and GMAN on the PeMSD4 dataset. For

TABLE VI
COMPUTATION TIME ON THE PEMSD4 DATASET

Model	Training (s/epoch)	Inference (s)
AGCRN	254.85	61.42
DCRNN	394.67	27.09
GMAN	230.55	11.41
Bi-STAT	239.21	13.33

the comparison methods, we use their official code implementations. All methods are run on a server with Intel[2] Xeon[2] E-2288G CPU and an NVIDIA Corporation TU102GL GPU. It can be observed that AGCRN and DCRNN cost more computation time for both training and inference due to the time-consuming process of the RNNs. In contrast, the transformer-based methods, i.e., GMAN and Bi-STAT (ours), show the substantially improved efficiency owning to the parallel and acceleration of the transformer architecture. Although Bi-STAT costs a slightly longer time than GMAN, it endows the model with the adaptive and dynamic computation capability (by the well-devised DHM module). Therefore, Bi-STAT yields more accurate predictions than GMAN, e.g., 29.96 versus 31.06 (RMSE).

V. CONCLUSION

In this article, we propose a Bi-STAT to deal with the urban traffic forecasting problem. Specifically, the model design is motivated by two intrinsic properties of the problem that is overlooked by existing methods. First, the imbalanced complexities of diverse forecasting problems require an adaptive computation model. Thus, we adopt the recurrent mechanism with a novel DHM to parsimoniously assign the essential computation load for each task. Second, the recollection of past traffic conditions plays an important role in the prediction of future traffic conditions. Therefore, we devise a recollection decoder to reconstruct the past traffic streams, which can provide extra context information for the future prediction task and serve as a regularizer to prevent the future prediction task from fitting aggressively. All the designed modules are well incorporated into our Bi-STAT model. The experiments demonstrate the effectiveness of each model component, and our Bi-STAT model outperforms all the state of the arts with a large margin.

However, modeling the complicated and complex spatial-temporal correlations heavily relies on the delicate design of the task-specific neural network, which requires substantial domain knowledge and abundant expert efforts. The recent surge of automated neural architecture search [49], [50] can provide a promising alternative to the application of the urban traffic forecasting task, which can automatically search for an optimal architecture to capture the spatial-temporal representation in the traffic network. The research emphases include the definition of the search space in consideration of the spatial-temporal correlations, as well as the devising of strategies to learn the network parameters related to traffic graphs, as the future direction of automatic urban traffic forecasting.

[2]Registered Trademark.

REFERENCES

[1] H. Yuan and G. Li, "A survey of traffic prediction: From spatio-temporal data to intelligent transportation," *Data Sci. Eng.*, vol. 6, no. 1, pp. 63–85, Mar. 2021.

[2] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *Proc. 28th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2019, pp. 1–7.

[3] X. Wang *et al.*, "Traffic flow prediction via spatial temporal graph neural network," in *Proc. Web Conf.*, 2020, pp. 1082–1092.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–14.

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

[6] 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. 1–12.

[7] 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.

[8] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[9] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "DNN-based prediction model for spatio-temporal data," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2016, pp. 1–4.

[10] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1–7.

[11] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 1, 2019, pp. 922–929.

[12] M. Xu *et al.*, "Spatial-temporal transformer networks for traffic flow forecasting," 2020, *arXiv:2001.02908*.

[13] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng, "GeoMAN: Multi-level attention networks for geo-sensory time series prediction," in *Proc. IJCAI*, vol. 2018, 2018, pp. 3428–3434.

[14] C. Zheng, X. Fan, C. Wang, and J. Qi, "Gman: A graph multi-attention network for traffic prediction," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 1, 2020, pp. 1234–1241.

[15] K. Chen, G. Chen, D. Xu, L. Zhang, Y. Huang, and A. Knoll, "NAST: Non-autoregressive spatial-temporal transformer for time series forecasting," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 1684–1694.

[16] S. L. Mullally and E. A. Maguire, "Memory, imagination, and predicting the future: A common brain mechanism?" *Neuroscientist*, vol. 20, no. 3, pp. 220–234, Jun. 2014.

[17] D. L. Schacter, D. R. Addis, D. Hassabis, V. C. Martin, R. N. Spreng, and K. K. Szpunar, "The future of memory: Remembering, imagining, and the brain," *Neuron*, vol. 76, no. 4, pp. 677–694, Nov. 2012.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[19] L. Zhu, Z. Xu, and Y. Yang, "Bidirectional multirate reconstruction for temporal modeling in videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2653–2662.

[20] W. W. Wei, "Time series analysis," in *The Oxford Handbook of Quantitative Methods in Psychology*, vol. 2. 2006.

[21] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results," *J. Transp. Eng.*, vol. 129, no. 6, pp. 664–672, Nov. 2003.

[22] E. Zivot and J. Wang, "Vector autoregressive models for multivariate time series," in *Modeling Financial Time Series With S-Plus*. 2006, pp. 385–429.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1097–1105.

[24] P. Jiang, F. Liu, and Y. Song, "A hybrid forecasting model based on date-framework strategy and improved feature selection technology for short-term load forecasting," *Energy*, vol. 119, pp. 694–709, Jan. 2017.

[25] S. Qin, F. Liu, C. Wang, Y. Song, and J. Qu, "Spatial-temporal analysis and projection of extreme particulate matter (PM10 and PM2.5) levels using association rules: A case study of the Jing-Jin-Ji region, China," *Atmos. Environ.*, vol. 120, pp. 339–350, Nov. 2015.

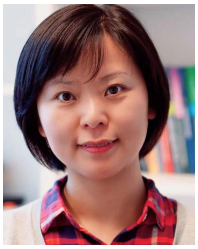
[26] D. Zhang, L. Yao, K. Chen, S. Wang, X. Chang, and Y. Liu, "Making sense of spatio-temporal preserving representations for EEG-based human intention recognition," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 3033–3044, Jul. 2020.
[27] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[28] Y. Xu, Y. Han, R. Hong, and Q. Tian, "Sequential video VLAD: Training the aggregation locally and temporally," *IEEE Trans. Image Process.*, vol. 27, no. 10, pp. 4933–4944, Oct. 2018.
[29] X. Chen, S. Chen, J. Yao, H. Zheng, Y. Zhang, and I. W. Tsang, "Learning on attribute-missing graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 2, pp. 740–757, Feb. 2020.
[30] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[31] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, and J. Zhang, "Urban traffic prediction from spatio-temporal data using deep meta learning," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Mining*, 2019, pp. 1720–1730.
[32] C. Zhang and P. Patras, "Long-term mobile traffic forecasting using deep spatio-temporal neural networks," in *Proc. 18th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jun. 2018, pp. 231–240.
[33] H. Yuan, G. Li, Z. Bao, and L. Feng, "An effective joint prediction model for travel demands and traffic flows," in *Proc. IEEE 37th Int. Conf. Data Eng. (ICDE)*, Apr. 2021, pp. 348–359.
[34] B. Liao *et al.*, "Deep sequence learning with auxiliary information for traffic prediction," in *KDD*, 2018.
[35] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Q. Z. Sheng, "STG2Seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 1981–1987.
[36] C. Song, Y. Lin, S. Guo, and H. Wan, "Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 1, 2020, pp. 914–921.
[37] J. Ye, J. Zhao, K. Ye, and C. Xu, "How to build a graph-based deep learning architecture in traffic domain: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 5, pp. 3904–3924, May 2022.
[38] M. Li and Z. Zhu, "Spatial-temporal fusion graph neural networks for traffic flow forecasting," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 5, 2021, pp. 4189–4196.
[39] Y. Chen, I. Segovia, and Y. R. Gel, "Z-GCNETs: Time zigzags at graph convolutional networks for time series forecasting," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 1684–1694.
[40] 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*, 2020, pp. 753–763.
[41] J. Zhang, Q. Wang, and Y. Han, "Multi-modal fusion with multi-level attention for visual dialog," *Inf. Process. Manage.*, vol. 57, no. 4, Jul. 2020, Art. no. 102152.
[42] A. Dosovitskiy *et al.*, "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–22.
[43] M. Caron *et al.*, "Emerging properties in self-supervised vision transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2021, pp. 9650–9660.
[44] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman, "Video action transformer network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 244–253.
[45] L. Cai, K. Janowicz, G. Mai, B. Yan, and R. Zhu, "Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting," *Trans. GIS*, vol. 24, no. 3, pp. 736–755, Jun. 2020.
[46] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 855–864.
[47] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *Stat*, vol. 1050, p. 21, 2016.
[48] C. Chen, K. Petty, A. Skabardonis, P. Varaiya, and Z. Jia, "Freeway performance measurement system: Mining loop detector data," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1748, no. 1, pp. 96–102, Jan. 2001.
[49] P. Ren *et al.*, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 1–34, May 2021.
[50] M. Zhang *et al.*, "One-shot neural architecture search: Maximising diversity to overcome catastrophic forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 9, pp. 2921–2935, Sep. 2020.

Changlu Chen received the B.S. and M.S. degrees in electronic information science and technology from Lanzhou University, Lanzhou, China. She is currently pursuing the Ph.D. degree with the University of Technology Sydney, Sydney, NSW, Australia.

Her current research interests include spatial-temporal learning and traffic forecasting.

Yanbin Liu received the B.E. and M.S. degrees from Tianjin University, Tianjin, China, in 2013 and 2015, respectively, and the Ph.D. degree from the University of Technology Sydney, Sydney, NSW, Australia, in 2021.

He is currently a Research Fellow with the Centre for Medical Research, The University of Western Australia, Perth, WA, Australia. His research interests lie in machine learning and deep learning for computer vision problems, especially learning with limited labeled data.



Ling Chen (Senior Member, IEEE) received the Ph.D. degree from Nanyang Technological University, Singapore, in 2008.

She was a Post-Doctoral Research Fellow with the L3S Research Center, University of Hannover, Hanover, Germany. She is currently an Associate Professor with the Centre for Artificial Intelligence, University of Technology Sydney, Sydney, NSW, Australia. Her papers appear in major conferences and journals, including SIGKDD, ICDM, SDM, and ACM TOIS. Her research interests include data mining, machine learning, social network analysis, and recommender systems.

Chengqi Zhang (Senior Member, IEEE) received the Ph.D. degree from The University of Queensland, Brisbane, QLD, Australia, in 1991, and the D.Sc. (Higher Doctorate) degree from Deakin University, Geelong, VIC, Australia, in 2002.

Since 2001, he has been a Professor of information technology with the University of Technology Sydney (UTS), Sydney, NSW, Australia, where he has been the Executive Director of UTS Data Science since 2016. He has published more than 200 research papers, including several in first-class international journals, such as *Artificial Intelligence* and the IEEE and ACM TRANSACTIONS. He has published six monographs, edited 16 books, and attracted 11 Australian Research Council grants. His research interests mainly focus on data mining and its applications.

Dr. Zhang has been the Chairman of the Australian Computer Society, National Committee for Artificial Intelligence, since 2005.