Masked Token Enabled Pre-training: A Task-Agnostic Approach for Understanding Complex Traffic Flow

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

08-02-2022 / 10-02-2022

CITATION

Hou, Lu; Geng, Yunxin; Han, Lingyi; Yang, Haojun; Zheng, Kan; Wang, Xianbing (2022): Masked Token Enabled Pre-training: A Task-Agnostic Approach for Understanding Complex Traffic Flow. TechRxiv. Preprint. <https://doi.org/10.36227/techrxiv.19134854.v1>

DOI

[10.36227/techrxiv.19134854.v1](https://doi.org/10.36227/techrxiv.19134854.v1)

Masked Token Enabled Pre-training: A Task-Agnostic Approach for Understanding Complex Traffic Flow

Lu Hou *Member, IEEE*, Yunxin Geng, Lingyi Han, Haojun Yang *Member, IEEE*, Kan Zheng *Senior Member, IEEE*, and Xianbin Wang *Fellow, IEEE*

Abstract—The conventional deep learning model performs well for traffic flow analysis by training with a large number of labeled data using a one-model-for-one-task approach, leading to huge computational complexity in dynamic intelligent transportation system (ITS) applications. To overcome this limitation, this paper propose a Token-based Self-Supervised Network (TSSN), which can learn TF features in a task-agnostic way, and provide a well bootstrapped pre-training model for a variety of tasks. TSSN tokenizes TF data into segments, each of which is named as a *token* and comprised of numerous consecutive points. Masked Token Prediction (MTP), a pretext task, is designed to understand the TF correlations by forecasting tokens that are randomly masked. MTP enables TSSN to capture the high-level intrinsic semantics of TF, and provide general-purpose token embeddings. Therefore, TSSN can be more generalized while keeping high performance. As a result, by replacing the final fully-connected layer with a set of untrained new layers and fine-tuning with small-scale task-specific data, TSSN can be deployed for a variety of downstream tasks. The simulation results demonstrate that the TSSN can improve overall performance on various downstream tasks when compared to state-of-the-art models.

Index Terms—Traffic flow, Tokenization, Self-supervised learning, Pre-training

I. INTRODUCTION

Time-series data analytics have shown considerable potentials in real-world applications such as intelligent traffic systems (ITS), smart healthcare and financial management. In ITS, traffic flow (TF) data, which represents the number of vehicles passing a fixed point within a given time interval, can accurately indicate the transportation situation and support making appropriate traffic route planning and congestion management. Therefore, the precise TF data analysis is critical to improve the reliability and efficiency of ITS applications [1].

Non-linearity and unpredictability are the key features of TF data that make them extremely challenging to process [2]. Meanwhile, TF data may display a wide variety of patterns depending on a number of factors. For example, due to the prevalence of roadside structures, TF may show specific trends in different blocks of the city [3]. Among all these patterns, discovering the traffic patterns during particular periods of time

in different days, i.e. finding the traffic temporal correlations, are the focus of this work. One main issue of TF data analysis is that the training of a specific model in machine learning is closely related to a given task. When the purpose is altered or replaced, the trained model may become obsolete, mandating the training of a new model from scratch. Another difficulty is that a large amount of labeled data is required to train a deep learning (DL) model, which could take a significant period of time and make the learning result less useful [4].

The advancement of self-supervised learning (SSL) has been shown to be successful and efficient in both natural language processing (NLP) and computer vision (CV) [5]. SSL could use unlabeled data and agnostic tasks to train a model. Then, through fine-tuning on small-scale task-specific data, downstream tasks, which are the ultimate tasks to solve in TF analysis, can attain high performance [6]. Bidirectional encoder representations from transformers (BERT), for example, can learn token correlations in natural language from two pre-training tasks and receive general token representations. Inspired by BERT, tokens and *sentences* can be created from the TF data, and subsequent processing can be applied to each token instead of each TF point. Even with the transformer’s attention mechanism, SSL models based on individual points would focus more on short-term correlation and high-frequency changes, potentially resulting in poor downstream task performance due to low generalization. To the best of our knowledge, none of the studies have attempted to treat several points of TF data as a whole.

Therefore, this paper presents a Token-based Self-Supervised Network (TSSN) which builds a pre-training model that can be used to bootstrap a variety of task-specific fine-tuning models. The TF data is divided into tokens, each of which contains a series of consecutive points. As a result, the model can acquire high-level contextual semantics and encode tokens with representations that leverage fundamental structures of TF data by considering multiple points as a token. Then, a masked tokens prediction (MTP) pretext task is defined as one in which a subset of tokens is masked at random and the pre-training model must guess these tokens from contextual tokens in a sentence. The token-based method may explore more macroscopic and long-term aspects than point-based methods, whilst the MTP blurs the lines between different tasks, allowing a bootstrapping model to learn the TF semantics for improving the task-specific model performance. Finally, a general model is learned, and specific tasks can

Lu Hou, Yunxin Geng, and Kan Zheng are with the Intelligent Computing and Communications (IC²) lab, Key Lab of Universal Wireless Communications, Ministry of Education, Beijing University of Posts and Telecommunications, China.

Lingyi Han is with the Big Data Center, China Aero Geophysical Survey & Remote Sensing Center for Natural Resources (AGRS).

Haojun Yang and Xianbin Wang are with the Department of Electrical and Computer Engineering, Western University, London, ON N6A 5B9, Canada.

be completed with minor fine-tuning. The pre-training model generates representations of each token for various downstream tasks. Our main contributions in this paper are presented as follows, i.e.,

- 1) A novel network, i.e. TSSN, is proposed for generating an effective task-agnostic model for various downstream tasks on TF data. Meanwhile, a novel pretext task, i.e. MTP, is designed to provide strong surrogate supervision signals for the pre-training of TSSN. The goal is to understand the temporal correlations of unlabeled TF data through tokenization and SSL.
- 2) Three types of downstream tasks, i.e. TF classification, prediction and completion, are solved by using the representations of tokens created in pre-training model. Only the final few layers are reconstructed and fine-tuned with task-specific and labeled data.
- 3) A vast number of experiments are conducted to compare TSSN to some traditional methods, as well as pre-training models with different numbers of points in a token. All three types of downstream tasks are evaluated, each of which includes different parameter configurations. The results demonstrate the effectiveness of the proposed TSSN clearly.

The rest contents of the paper are organized as follows. Section II explores the state-of-the-art of SSL on time series data. The TF data are processed in Section III, and the pre-training and fine-tuning tasks are discussed in Section IV. A lot of experiments are conducted in Section V with detailed analysis. Section VI concludes this paper.

II. RELATED WORKS

Understanding the semantic-level properties of TF is required for the establishment of ITS applications. For TF analysis, tensor-based methods have been used, but their high complexity and lack of scalability make further development difficult [7]. As a result, DL models have been considered feature learners in a number of ways. These cutting-edge models are primarily concerned with boosting performance by strengthening model structure [8] or merging multi-modal data [9]. However, none of the literature attempts to learn the features by tokenizing the TF in the same way that words are tokenized in the NLP field.

A. Traffic flow prediction

The TF prediction is crucial for a variety of ITS applications. The majority of studies use historical data to anticipate future TF utilizing tensor-based approaches or DL models. The impact of disruptions on TF prediction is investigated by Zheng *et al.* [10]. They propose two factors: inherent influence factor and disturbance influence factor, which they eliminate from TF data to represent the inherent patterns. Feng *et al.* [11] propose an supported vector machine (SVM)-based TF prediction model that combines spatial and temporal correlations, whereas Han *et al.* [12] use an long-short term memory (LSTM) network to make predictions based on representation learning of shape-based features. Both of these studies concentrate on predictions that are made in the near period (less than half an hour). For

both short-term and long-term TF predictions, Huang *et al.* [13] present a graph convolutional networks (GCN)-based model (from 5 min to 60 min). By using spatial features, the model can achieve a high level of performance.

B. Traffic flow classification

Other groups emerge from the TF data when different contextual circumstances, such as weather, accidents, weekends, or vacations, are considered. Cinsdikici *et al.* [14] offer a two-phase approach to capture TF density variations and classify TF patterns at each time instant. The TF can be split into three categories: free flow, the dense flow and the congested flow. Bui *et al.* [15] investigate how traffic sound data can be used to classify the traffic density. To extract sound features and achieve TF categorization, they propose a graph-based representation learning approach employing convolutional neural networks (CNN). The idea is to distinguish rush hour at different times of the day (morning or evening).

C. Traffic flow completion

Tensor completion is used to model the TF data using high-dimension tensors and generate high-order decomposition to get low-rank representations, and then solve the completion problem using norm regularization [16], [17]. However, due to their considerable complexity, these strategies are difficult to put into practice. The neural network-based TF data completion is more efficient and can even fill in long stretches of missing data. Li *et al.* [18] investigate how generative adversarial networks (GAN) might be used to fill in missing TF data in a graph-based approach. Han *et al.* [19] offer a GAN-based data completion method that combines tensor modeling to achieve long-term TF completion. The main idea is to use low-dimension representations to recover the most fitting TF data sequences. The results reveal that the performance is satisfactory even for one-week data completion.

D. Self-supervised learning on time series

SSL is gaining popularity as a tool for extracting features from unlabeled data. SSL has a lot of potential in NLP thanks to transformer structure. One of the most successful models is Google's BERT, which employs transformer encoders to learn sentence representations [20]. In BERT, tokens are masked at random, and two pre-training tasks, masked language modeling (MLM) and next sentence prediction (NSP), are introduced during pre-training stage.

Yuan *et al.* [21], inspired by BERT, develop a self-supervised pre-training model that gains general knowledge from large-scale unlabeled satellite image time series and applies it to classification tasks with scarce-label data. Ma *et al.* [22] present a time series clustering approach in which the pseudo-class labels are generated through k -means algorithm. As a result, clustering can be accomplished without a huge amount of labeled data. Shi *et al.* [23] devise two new acoustic time series classification pre-training tasks. The proposed model can be utilized to increase performance on tiny labeled datasets.

According to the current state of the art, TF data processing usually treats each point as a single basic unit rather than a

Fig. 1. Illustration of the whole structure.

series of points. Another issue is that current solutions are only targeted at specific activities. For instance, the model suggested in [13] can only make predictions on TF data, whereas [19] is only for TF completion. Few studies have attempted to develop a task-agnostic technique that can be applied to various aspects of ITS applications using SSL. Therefore, employing token-based processing and pre-training methods, this work primarily offers an SSL model that can provide sufficient bootstrapping for various downstream tasks.

III. TRAFFIC FLOW DATA PRE-PROCESSING

Let $\mathcal{X}$ denote the dataset which contains traffic flow data of several days on multiple road segments. Assume there is totally N points for one day. All points in a day are treated as a sentence. Let $x_i \in \mathcal{X}, i = 1, \dots, N$ denote the i^{th} point in a sentence. Each sentence contains several tokens. Let K denote the number of points in a token. Assume that K is a factor of N . Then, the number of tokens in one sentence is $M = N/K$. Let $\mathbf{t}_j, j \in [1, M]$ denote the j^{th} token in one sentence, then we have $\mathbf{t}_j := \mathbf{x}_{(j-1)K:jK} = \{x_{(j-1)K+1}, x_{(j-1)K+2}, \dots, x_{jK}\}$. The rationale is to treat consecutive K points as a unit and learn the features or patterns that it possesses.

The TF data are normalized before training. Each point is scaled into $[0, 1]$ by min-max normalization as follow,

$$x_i \leftarrow \frac{x_i - \min(\mathcal{X})}{\max(\mathcal{X}) - \min(\mathcal{X})}. \quad (1)$$

A part of tokens is masked with random numbers to let the model learn the representations in self-supervised manner. All points in masked tokens are added by a positive or negative random value, i.e.,

$$\hat{\mathbf{t}}_j = \begin{cases} \{x_{(j-1)K+1} + \delta, x_{(j-1)K+2} + \delta, \dots, x_{jK} + \delta\}, \xi_j < \alpha, \\ \{x_{(j-1)K+1}, x_{(j-1)K+2}, \dots, x_{jK}\}, \xi_j \geq \alpha, \end{cases} \quad (2)$$

where ξ_j is the j^{th} random number that follows $\mathcal{U}(0, 1)$, and α stands for the masking ratio. For η probability, the masking value δ is sampled from $\mathcal{U}(0, b)$. Otherwise, $\delta \sim \mathcal{U}(-b, 0)$. Let

TABLE I
MATHEMATICAL NOTATIONS

Notations of General Symbols and Operations			
Notation		Definition	
$\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$		A matrix of $I_1 \times I_2$ dimensions	
$\mathbf{x} \in \mathbb{R}^I$		A vector of I dimension	
x_{i_1, i_2}, x_i		An element of a matrix/vector	
$\mathbf{x}_{i_1:i_2}$		Elements between index i_1 (not included) and i_2 (included)	
$\mathbf{X}^{\text{F}(I)}$		Flatten a matrix to I -dimension vector	
$\mathbb{S}(\mathbf{z})$		Softmax on the input $\mathbf{z}$	
$[\mathbf{X}_1, \mathbf{X}_2, \dots]_{\&(i)}$		Concatenate over the i -th dimension	
$\mathbb{L}\{\cdot\}$		Layer normalization	
$\mathbb{E}\{\cdot\}$		Expectation of the input	
$g(\cdot)$		Activation function	
$\odot$		Element-wise multiplication	
Notations of TSSN			
Notation	Definition	Notation	Definition
$\mathcal{X}$	Dataset	N	Number of TF data in a day
K	Token length	M	Number of tokens in a day
$\mathbf{x}$	A series of TF points	$\mathbf{t}$	A sequence of TF tokens
η	Proportion of positive/negative masking value	$\mathcal{U}$	Uniform distribution
Y	Number of encoders	D	Embedding dimension
α	Masking ratio	δ	Masking value
i	Index of point	j	Index of token
d	Index of embedding	$\mathbf{m}$	Masking position
$\mathbf{P}$	Positional encoding	S	Number of categories
P	Length of prediction horizon	C	Number of completion data

$\mathbf{m}$ record whether the element of a specific position in input sequence is masked. Then, it can be defined as follow,

$$\mathbf{m}_{(j-1)K:jK} = \begin{cases} \underbrace{\{1, 1, \dots, 1\}}_K, \xi_j < \alpha, \\ \underbrace{\{0, 0, \dots, 0\}}_K, \xi_j \geq \alpha. \end{cases} \quad (3)$$

It's worth noting that instead of random points, the entire token is masked with the same value, which is crucial to the TSSN's success.

IV. PRE-TRAINING OF TSSN AND FINE-TUNING

The embedding layer, positional encoding (PE) layer and numerous Transformer encoders make up the majority of the proposed model.

A. Model Structure

Assume each input is denoted as $\mathbf{x} \in \mathbb{R}^N$. It is transformed into matrix as $\mathbf{X} \in \mathbb{R}^{M \times K}$, which means M tokens with K points in each token.

1) *Embedding*: Each token $\mathbf{t}$ is embedded to a D -dimension vector as $\bar{\mathbf{t}}$ by a fully-connected layer. The parameters are trainable to improve the precision of embedding during the learning process. Therefore, the token embedding can be given as follow,

$$\bar{\mathbf{X}} = \mathbf{X} \mathbf{W}^{\text{Embedding}}, \quad (4)$$

where $\mathbf{W}^{\text{Embedding}} \in \mathbb{R}^{K \times D}$, and the dimension of $\bar{\mathbf{X}}$ becomes $M \times D$.

2) *Positional encoding*: The positional information of input sequence is encoded by a pair of sin and cos functions with different frequencies [24], i.e.,

$$p_{j,d} = \begin{cases} \sin(j/10000^{d/D}), & \text{if } d \bmod 2 = 0, \\ \cos(j/10000^{d/D}), & \text{if } d \bmod 2 = 1, \end{cases} \quad (5)$$

where $j \in [1, M]$ and $d \in [1, D]$. The $p_{j,d}$ is add to the $\bar{\mathbf{X}}$ element-wisely, i.e.,

$$\bar{x}_{j,d} \leftarrow \bar{x}_{j,d} + p_{j,d}, \forall j \in [1, M], d \in [1, D], \quad (6)$$

It's worth noting that each point in a token must have the same PE value, implying that each point has the same impact on tokens in other locations. As a result, PEs include token-level positional information rather than point-level positional information.

3) *Transformer Encoder*: The $\bar{\mathbf{X}}$ is then processed by Y transformer encoders with L -head attention, where $D/L \in \mathbb{Z}$. $\bar{\mathbf{X}}$ is transformed by fully-connected layers into L query, key and value matrices, each of which is denoted as $\mathbf{Q}_l$, $\mathbf{K}_l$ and $\mathbf{V}_l$, $l \in [1, L]$, respectively. Let $n = D/L$, $\mathbf{Q}_l$, $\mathbf{K}_l$ and $\mathbf{V}_l$ can be calculated as follows,

$$\mathbf{Q}_l = \bar{\mathbf{X}} \mathbf{W}_l^Q, \mathbf{K}_l = \bar{\mathbf{X}} \mathbf{W}_l^K, \mathbf{V}_l = \bar{\mathbf{X}} \mathbf{W}_l^V, \quad (7)$$

where $\mathbf{W}_l^Q, \mathbf{W}_l^K, \mathbf{W}_l^V \in \mathbb{R}^{D \times n}$ and $\mathbf{Q}_l, \mathbf{K}_l, \mathbf{V}_l \in \mathbb{R}^{M \times n}$, respectively.

The l^{th} attention head $\mathbf{H}_l$ can be calculated as follow,

$$\mathbf{H}_l = \mathbb{S}(\frac{\mathbf{Q}_l \mathbf{K}_l^\top}{\sqrt{n}}) \mathbf{V}_l, \quad (8)$$

where $\mathbf{H}_l \in \mathbb{R}^{M \times n}$. Then, the multi-head attention is calculated by the concatenation of L heads with linear projections by a fully-connected layer, i.e.,

$$\mathbf{A} = [\mathbf{H}_1, \mathbf{H}_2, \dots, \mathbf{H}_L]_{\&2} \mathbf{W}^O, \quad (9)$$

where $\mathbf{W}^O \in \mathbb{R}^{D \times D}$ and the dimension of $\mathbf{A}$ is equivalent to $\bar{\mathbf{X}}$.

Two-layer position-wise feed-forward networks are applied after $\bar{\mathbf{X}}$ to get the embedding representation of tokens. Besides, residual connection and layer normalization are applied for better convergence, according to [24]. Let $\bar{\mathbf{X}} \leftarrow \mathbb{L}(\bar{\mathbf{X}} + \mathbf{A})$, then the representations can be calculated as follow,

$$\bar{\mathbf{Y}} = g(\bar{\mathbf{X}} \mathbf{W}_1 + \mathbf{b}_1) \mathbf{W}_2 + \mathbf{b}_2, \quad (10)$$

where g represents the activation function, $\mathbf{W}_1 \in \mathbb{R}^{D \times D^{\text{hid}}}$, $\mathbf{b}_1 \in \mathbb{R}^{D^{\text{hid}} \times 1}$ and $\mathbf{W}_2 \in \mathbb{R}^{D^{\text{hid}} \times D}$, $\mathbf{b}_2 \in \mathbb{R}^{D \times 1}$.

The *encoded representations of each token in a sentence* are the transformer encoders' outputs $\bar{\mathbf{Y}}$. As a result, the layer is known as the TSSN *representation layer*. To retrieve the final sequence of TF data, instead of using transformer decoders, a fully-connected layer is utilized to reduce the dimension and flatten the vector, i.e.,

$$\mathbf{y} = (\bar{\mathbf{Y}} \mathbf{W}_3 + \mathbf{b}_3)^{\text{F}(N)}, \quad (11)$$

where $\mathbf{W}_3 \in \mathbb{R}^{D \times K}$, $\mathbf{b}_3 \in \mathbb{R}^{K \times 1}$, and the final dimension of $\mathbf{y}$ remains the same with the premier input $\mathbf{x}$.

B. Pre-training

The pre-training stage and the fine-tuning stage are the two stages of the training process. The pre-training stage enables TSSN to extract implicit token features in a task-independent manner using MTP, which involves predicting tokens that are randomly masked. As a result, the mean square error (MSE) between the predicted data and the true data that are masked is used as the pre-training loss function, i.e.,

$$J^{\text{pre}}(\theta) = \mathbb{E}_{\mathbf{t} \sim \mathcal{X}} \left\{ \sum_{j=1}^M [\mathbf{m}_j \odot (\mathbf{y}_{(j-1)K:jK} - \mathbf{t}_j)]^2 \right\}, \quad (12)$$

where θ stands for all trainable parameters of the model. Only masked tokens are considered in the loss calculation, according to eq (12). TSSN can learn more abstract and long-term features because the masked tokens are chosen at random and consist of numerous consecutive points,

C. Fine-tuning

To fine-tune TSSN for various downstream tasks, the TSSN's final fully-connected layer is replaced by new layers with untrained parameters. For the fine-tuning tasks, the pre-training model only produces the embedding representations $\bar{\mathbf{Y}}$ of each token in a sentence. With task-specific labeled data, a little further training is added. Furthermore, the loss function is determined by the downstream tasks' purpose. Only the parameters of posterior layers are updated during the fine-tuning process, while the parameters in pre-training model stay intact.

To evaluate the performance of TSSN, three types of downstream tasks are considered: TF classification, prediction and completion.

1) *Traffic flow classification*: For S -category classification tasks, two fully-connected layers are attached after $\bar{\mathbf{Y}}$, i.e.,

$$\mathbf{y}^C = \mathbb{S}\{g(\bar{\mathbf{Y}} \mathbf{W}_1^C + \mathbf{b}_1^C) \mathbf{W}_2^C\}, \quad (13)$$

where $\mathbf{W}_1^C \in \mathbb{R}^{D \times D^{\text{C-hid}}}$, $\mathbf{b}_1^C \in \mathbb{R}^{D^{\text{C-hid}} \times 1}$, and $\mathbf{W}_2^C \in \mathbb{R}^{D^{\text{C-hid}} \times S}$. The final output $\mathbf{y}^C$ becomes a S -dimension vector, each element y_s^C of which represents the probability of being classified as the s^{th} category.

The labeled dataset for classification task is denoted as $\mathcal{X}^C = \{(\mathbf{x}, \boldsymbol{\rho})\}$, where $\boldsymbol{\rho} := [\rho_1, \rho_2, \dots, \rho_S]$, $\rho_1, \rho_2 \dots \rho_S \in \{0, 1\}$ and $\sum_{s=1}^S \rho_s = 1$. $\boldsymbol{\rho}$ stands for the one-hot encodings

of categorical labels where $\rho_s = 1$ means $\mathbf{x}$ belong to the category s . Then, the loss function is defined by the cross entropy between the predicted labels and the true labels, i.e.,

$$J^C = -\mathbb{E}_{\mathbf{x} \sim \mathcal{X}} \left\{ \sum_{s=1}^S \rho_s \log y_s^C \right\}. \quad (14)$$

2) *Traffic flow prediction*: To make precise predictions on the future horizon of TF based on existing data, TSSN is re-constructed by connecting two dense layers after the representation layer, i.e., for P -horizon prediction task, the output $\mathbf{y}^P$ is given as follow,

$$\mathbf{y}^P = g(\bar{\mathbf{Y}}\mathbf{W}_1^P + \mathbf{b}_1^P)\mathbf{W}_2^P + \mathbf{b}_2^P, \quad (15)$$

where $\mathbf{W}_1^P \in \mathbb{R}^{D \times D^{\text{P-hid}}}$, $\mathbf{b}_1^P \in \mathbb{R}^{D^{\text{P-hid}} \times 1}$ and $\mathbf{W}_2^P \in \mathbb{R}^{D^{\text{P-hid}} \times P}$, $\mathbf{b}_2^P \in \mathbb{R}^{P \times 1}$, respectively. The dimension of output is equivalent to the length of horizon that required to be forecasted. The loss function is defined as the MSE between the predicted values and the true values, i.e.,

$$J^P = -\mathbb{E}_{\mathbf{x} \sim \mathcal{X}} \{(\mathbf{y}^P - \mathbf{x}^P)^2\}, \quad (16)$$

where $\mathbf{x}^P$ is the true values of the predicted horizon.

3) *Traffic flow data completion*: The TF data completion task fills in the blanks with existing TF values. The positions of missing data are sequentially chosen with randomness. Assuming that the total missing number is C , the task's fine-tuning model is as follow, i.e.,

$$\mathbf{y}^{\text{Cm}} = g(\bar{\mathbf{Y}}\mathbf{W}_1^{\text{Cm}} + \mathbf{b}_1^{\text{Cm}})\mathbf{W}_2^{\text{Cm}} + \mathbf{b}_2^{\text{Cm}}, \quad (17)$$

where $\mathbf{W}_1^{\text{Cm}} \in \mathbb{R}^{D \times D^{\text{Cm-hid}}}$, $\mathbf{b}_1^{\text{Cm}} \in \mathbb{R}^{D^{\text{Cm-hid}}}$ and $\mathbf{W}_2^{\text{Cm}} \in \mathbb{R}^{D^{\text{Cm-hid}} \times N}$. The output dimension is equivalent to the number of TF data in one sentence. However, the loss computation only considers the TF values at the missing positions. A fixed value β is used to replace the missing data. Assume that the missing places $\mathbf{m}^{\text{Cm}}$ are represented by a binary vector of N elements, each of which is specified as follow,

$$m_i = \begin{cases} 1, & \text{If } x_i \text{ is missing,} \\ 0, & \text{Otherwise.} \end{cases} \quad (18)$$

Then, the loss function can be defined using MSE as follow,

$$J^{\text{Cm}} = \mathbb{E}_{\mathbf{x} \sim \mathcal{X}} \{[\mathbf{m}^{\text{Cm}} \odot (\mathbf{y}^{\text{Cm}} - \mathbf{x})]^2\}. \quad (19)$$

In contrast to the pre-training tasks, missing data is substituted with a set value instead of a random number. Furthermore, unlike the prediction task, which merely outputs the numbers at expected positions, the output $\mathbf{y}^{\text{Cm}}$ is a whole sentence.

V. EXPERIMENTAL RESULTS AND ANALYSIS

A. Experiments setup

1) *Pre-training*: Five distinct values of K are evaluated to verify the performance of TSSN: $K = 1$ (**five-minute** TF value), $K = 2$ (**ten-minute** TF value), $K = 6$ (**half-hour** TF value), $K = 12$ (**one-hour** TF value) and $K = 24$ (**two-hour** TF value). $D = 64$ is the embedding dimension, which means that each token is represented by a 64-dimension vector. The source sentence embeddings are fed into three repeated transformer encoders. The number of simultaneous attention

TABLE II
LIST OF EXPERIMENTAL PARAMETERS

Parameter	Value	Description
K	1, 2, 6, 12, 24	Number of points in a token
D	64	Embedding dimension
M	288, 144, 48, 24, 12	Number of tokens in a day
L	8	Number of attention heads
B	128	Batch size
η	50 %	Proportion of positive/negative masking value
α	15 %	Masking rate
Y	3	Number of stacked encoders
b	0.5	Boundary of masking noise
β	0	Replace value for missing data
lr	10^{-4} for pre-training, 2×10^{-5} for fine-tuning	Learning rate

heads in each decoder is $L = 8$, and the query, key, and value vectors have a dimension of $D/L = 8$. As a result, the output dimension of the representation is $M \times 64$, where $M = 288, 144, 48, 24, 12$ for various values of K .

The datasets for pre-training are collected from Caltrans Performance Measurement System (PeMS) [25]. It has a large number of detectors installed throughout California's freeway system. TF data is collected every five minutes, which results $60/5 = 12$ points per minute and $N = 12 \times 24 = 288$ points per day. The original data are discrete values that have been smoothed with an 11-point window that averages each five points from before and after the current value. Pre-training uses a total of 3, 506, 846 points of TF data, with 80 % designated as training data and the remaining 20 % designated as validation data. The TF data are normalized according to eq.(1) before training. The pre-training data are randomly masked according to eq.(2), where the mask rate $\alpha = 15$ % and the boundary of masking value $b = 0.5$.

The TSSN is trained for 50 epochs with batch size 128 for each K value. Each epoch of training is followed by a validation process. The Adam optimizer is used, with an initial learning rate of 10^{-4} , which is warmed up throughout the first three epochs and then automatically and adaptively lowered. For all the fully-connected layers, the dropout rate is set at 0.1. The Gaussian error linear unit (GELU) activation function [26] is used throughout the whole TSSN architecture. All of the tests in this paper were run on NVIDIATM TeslaTM graphics processing units (GPUs) using compute unified device architecture (CUDA)TM 10 and PyTorch¹.

2) *Fine-tuning*: The initial learning rate is reduced to 2×10^{-5} during the fine-tuning step to make the learning parameters as close to the optimal value as possible. All the other hyper-parameters are unchanged from the pre-training stage. Besides, weekday-weekend classification, P -horizon data prediction and C -point data completion are three types of

¹<https://pytorch.org/>

TABLE III
PERFORMANCE OF TRAFFIC FLOW CLASSIFICATION TASK.

Metric	Accuracy	Precision	Recall	F1 score	Kappa
$K = 1$	88.55 %	0.938	0.932	0.929	0.826
$K = 2$	99.75 %	0.996	0.996	0.996	0.991
$K = 6$	87.35 %	0.932	0.925	0.921	0.806
$K = 12$	97.94 %	0.986	0.986	0.986	0.966
$K = 24$	96.39 %	0.979	0.979	0.979	0.948
FC	80.12 %	0.899	0.882	0.872	0.681
GRU	83.63 %	0.817	0.989	0.895	0.539

downstream tasks that are evaluated.

a) *Weekday-weekend classification*: One of the categorical downstream tasks of TSSN is the weekday-weekend classification task. Weekday and weekend TF data have distinct properties and patterns. The purpose is to classify the day as either a weekday or a weekend day according to the TF data. This task makes use of a new dataset called Seattle Inductive Loop Detector Dataset [27], [28]. The data are gathered in the same manner as PeMS. Meanwhile, each day is assigned the labels of weekday and weekend, with $\rho = [0, 1]$ denoting weekday and $\rho = [1, 0]$ denoting weekend. The hidden size is $D^{\text{C-hid}} = 64$. There are up to 100 fine-tuning epochs, each of which is followed by a validation phase. Two baselines are introduced, namely three-layer fully connected (FC) network and gated recurrent unit (GRU) network [29], to compare numerous metrics such as average classification accuracy, precision, recall, F1 score and Kappa coefficient.

b) *P-horizon prediction*: Short-term and long-term predictions are considered as downstream prediction tasks. For short-term predictions, P is set to be 1, 6, 12 which represent 5-min horizon, 10-min horizon and one-hour horizon predictions, while $P = 36$ (3-h horizon) and $P = 72$ (6-h horizon) stand for the long-term predictions, respectively. The hidden size $D^{\text{P-hid}}$ is set to 64, and the fine-tuning processes end only when the validation losses have stabilized. A bidirectional long-short term memory (Bi-LSTM) network [27] and a Transformer network [24] are among the comparative approaches. Through bidirectional feedback loops, Bi-LSTM is specifically built to capture patterns in sequence learning tasks. The metrics include mean absolute error (MAE), root mean square error (RMSE), mean absolute percentage error (MAPE) and R^2 .

c) *C-point data completion*: Under varying values of K , two tasks, namely 12-point (1 h) and 36-point (3 h) data completion, are tested. The missing values are replaced with $\beta = 0$, and the hidden layer's size is set to $D^{\text{Cm-hid}} = 64$. The baselines are a three-layer FC network and a Transformer network [24]. In addition, TSSN and a GAN-based model proposed in [19] are compared. The metrics are still the same as for prediction tasks.

B. Results and analysis

1) *Convergence*: All multiple epochs of training, all of the pre-training models converge to a low loss value and remain stable, as illustrated in Fig. 2(a). It is obvious that each token

with a smaller number of points (K value) has a lower loss value. This is due to the fact that a higher K value makes learning more difficult, but it can capture long-term temporal correlations.

After the first 20 epochs of training, the validation accuracies of all K values exceed 95 %, and when converged, they reach above 98 % for the classification task. The FC model takes longer to converge than any of the TSSN models, but the ultimate result is satisfactory. The GRU slowly converges to a high loss value, making it unsuitable for classifying weekdays and weekends.

Figure 2(c) and 2(d) show that for both prediction and completion tasks, each model can converge to a low loss value, although the converge speeds are different.

2) *Weekday-weekend classification*: Table III displays the classification task's results, where TSSN($K = 2$) achieves the highest accuracy. All TSSNs outperform FC and GRU networks in terms of accuracy, implying that TSSN can greatly improve classification task performance. The Kappa coefficients also suggest that all of the TSSNs are valid and dependable, whereas the classification consistency for FC network and GRU network can only be considered sufficient and moderate, respectively.

3) *Traffic flow prediction*: The literature generally consider that for TF data, $P \leq 12$ (1 h) stands for short-time prediction and $P > 12$ represents long-time prediction. MAPE = 20 % is generally considered as the boundary between model availability and unavailability. The results illustrated from Fig. 3 to Fig. 6 come from TF data of a week which is chosen at random.

a) *Short-time prediction*: Table IV shows that Bi-LSTM performs best for 1-horizon (5 min) prediction task. Over the MAPE metric, the highest performance of TSSN ($K = 12$) trails Bi-LSTM by around 61.22 %. This is due to the fact that Bi-LSTM are better at capturing short-term features, whereas TSSNs are better at capturing intermediate- and even long-term characteristics. Since the MAPE is approaching 20 %, the Transformer network, as well as the TSSN ($K = 1$), which only considers individual points and ignores the intrinsic correlations of consecutive points, are hardly capable. The same tendencies can be seen in Fig. 3, where Bi-LSTM fits the true data best, while all other models vary from the real values in some way.

On MAPE, TSSNs begins to outperform Bi-LSTM for 6-horizon (30 min) and 12-horizon (1 h) prediction tasks, demonstrating that TSSN can capture more accurate features than Bi-LSTM. This is because that TSSN can learn the representations of TF tokens. Thus, it can better extract TF internal patterns. However, it is not obvious that a higher K value equates to superior longer-horizon prediction ability. But in general, for more than one-horizon prediction tasks, multiple values of K outperform single values of K and the other two baselines.

b) *Long-time prediction*: The MAPEs of all the TSSNs from $K = 2$ to $K = 24$ are less than 20 % for the 36-horizon (3 h) prediction task. TSSN ($K = 12$) has the best performance among all of them. The TSSN ($K = 24$) slightly falls behind TSSN ($K = 12$) by 4.7 %. The MAPE of TSSN ($K = 1$) exceeds 20 % by 9.7 %, indicating that it is unavailable and demonstrating the utility of token-based pre-training model for

Fig. 2. Illustration of the convergence of different tasks.

Fig. 3. Illustration of the 1-horizon prediction results of seven days ($P = 1$).

Fig. 4. Illustration of the 6-horizon prediction results of seven days ($P = 6$).

long-term prediction tasks. For this task, the Bi-LSTM and Transformer models are completely disabled. Similar trends may also be seen in Fig. 6.

The prediction task with a 72-horizon (6 h) can be much more challenging than the 36-horizon task. The best-performing model is TSSN ($K = 12$), whose MAPE is just 0.5 % higher than the boundary. Meanwhile, over MAPE, TSSN ($K = 24$) falls behind TSSN ($K = 12$) by 0.8 %. The other models, particularly the two baselines, are insufficient for this task.

In Fig. 7, the TSSN ($K = 24$) is chosen to compare with TSSN ($K = 1$), Bi-LSTM and Transformer over several lengths of predicted horizon under varied metrics. The Bi-LSTM performs best at the start, i.e., $P = 1$. Then it degrades to the worst in terms of long-term prediction. In comparison to the Transformer model, the results of the TSSN ($K = 1$), which merely applies a pre-training processing, remain dismal at all lengths of prediction horizon. This suggests that the Transformer structure is not well suited for TF prediction

Fig. 5. Illustration of the 12-horizon prediction results of seven days ($P = 12$).

Fig. 6. Illustration of the 36-horizon prediction results of seven days ($P = 36$).

Fig. 7. Illustration of the prediction-task performances of different models.

tasks, and that **employing pre-training alone cannot improve prediction task performance**. Except for $P = 1$, the TSSNs ($K = 24$) perform the best. The results clearly suggest that the proposed token-based approaches can greatly improve prediction capabilities.

4) *Traffic flow completion*: Three typical patterns are identified for both 12-point and 36-point completion, including the *upslope pattern*, the *stationary pattern* and the *downslope pattern*. Each pattern represents the TF's increasing, stable, and

decreasing trends, respectively. To demonstrate the performance of the TF completion task, a random day is chosen. Table V shows that the overall completion performance of TSSN ($K = 24$) is the best among all the other models. The MAPEs of all TSSN models are less than 20 % for 12-point completion. However, because their MAPEs are below the availability boundary, only TSSN ($K = 12$) and TSSN ($K = 24$) can be used in 36-point completion task.

In [19], the MAPE of 36-point completion is slightly higher

TABLE IV
PERFORMANCE OF TRAFFIC FLOW PREDICTION

Task: 1-horizon prediction				
Metric	MAE	RMSE	MAPE	R^2
$K = 1$	3.7645	4.7540	18.42 %	0.9902
$K = 2$	2.3597	3.3410	8.36 %	0.9952
$K = 6$	3.1763	4.1776	9.61 %	0.9925
$K = 12$	2.7303	3.5384	7.11 %	0.9946
$K = 24$	2.9932	3.9030	11.04 %	0.9934
Bi-LSTM	1.7132	2.6243	4.41 %	0.9970
Transformer	5.5000	7.2667	17.91 %	0.9772
Task: 6-horizon prediction				
$K = 1$	4.9391	6.6847	20.69 %	0.9802
$K = 2$	4.2624	5.7358	15.00 %	0.9855
$K = 6$	4.4928	5.6759	23.14 %	0.9858
$K = 12$	3.4897	4.7138	10.75 %	0.9902
$K = 24$	3.5807	4.9743	9.61 %	0.9891
Bi-LSTM	3.7673	5.6619	11.74 %	0.9858
Transformer	5.6278	7.1548	23.35 %	0.9774
Task: 12-horizon prediction				
$K = 1$	6.9923	9.2378	30.84 %	0.9622
$K = 2$	4.6361	7.0126	14.33 %	0.9782
$K = 6$	3.8961	5.9076	10.04 %	0.9845
$K = 12$	4.6468	6.8840	10.88 %	0.9790
$K = 24$	4.6498	6.5917	13.05 %	0.9807
Bi-LSTM	5.9903	9.9959	12.18 %	0.9557
Transformer	10.5247	13.2723	38.97 %	0.9219
Task: 36-horizon prediction				
$K = 1$	9.0632	14.5817	21.94 %	0.9091
$K = 2$	7.6587	12.4783	18.39 %	0.9334
$K = 6$	8.4476	15.5894	19.20 %	0.8961
$K = 12$	6.9074	11.1830	15.53 %	0.9465
$K = 24$	7.3299	12.3964	16.26 %	0.9343
Bi-LSTM	12.4419	19.6025	28.13 %	0.8357
Transformer	11.6615	17.0404	43.08 %	0.8758
Task: 72-horizon prediction				
$K = 1$	11.3611	16.5521	31.02 %	0.9066
$K = 2$	10.2837	15.9426	22.50 %	0.8913
$K = 6$	9.8125	16.0190	25.91 %	0.8902
$K = 12$	8.7609	13.6979	20.10 %	0.9197
$K = 24$	9.0221	13.7188	20.26 %	0.9195
Bi-LSTM	23.0784	29.3615	84.25 %	0.6313
Transformer	20.1169	24.6235	88.81 %	0.7407

than 15 %, and the model is constructed by combining GAN and tensor-based completion approaches. The TSSN ($K = 24$) can achieve comparable results while enabling generality for additional types of downstream tasks.

Figure 8 shows that TSSN ($K = 24$) can fit the missing

TABLE V
PERFORMANCE OF TF DATA COMPLETION

Task: 12-point completion				
Metric	MAE	RMSE	MAPE	R^2
$K = 1$	9.9199	12.9459	18.68 %	0.9799
$K = 2$	9.7261	12.8552	17.11 %	0.9800
$K = 6$	6.0661	8.2628	15.96 %	0.9909
$K = 12$	5.9774	8.7689	14.21 %	0.9902
$K = 24$	4.8726	7.0772	12.09 %	0.9935
FC	10.8746	16.1315	24.55 %	0.9688
Transformer	15.4246	19.8795	29.11 %	0.9536
Task: 36-point completion				
$K = 1$	12.1165	18.0808	21.35 %	0.9751
$K = 2$	10.3310	15.5922	23.98 %	0.9825
$K = 6$	10.8910	15.6313	20.54 %	0.9826
$K = 12$	10.3309	15.5496	18.95 %	0.9851
$K = 24$	8.9518	14.0352	14.78 %	0.9859
FC	12.9212	19.3906	25.19 %	0.9551
Transformer	19.3298	25.6594	34.20 %	0.9218

values well for the *upslope pattern* of 12-point completion. Furthermore, TSSN ($K = 24$) outperforms all four methods on both *stationary pattern* and *downslope pattern*, albeit it still does not match the missing values optimally. This illustrates that TSSN can increase TF completion task accuracy.

The results of 36-point completion tasks reveal that all models can fit the missing values with a minor bias for both *upslope pattern* and *downslope pattern*, as shown in Fig. 9. TSSN ($K = 24$) matches the missing value optimum on the *upslope pattern*, but not well enough on the *downslope pattern*. This is caused by the randomness of the fine-tuning process. In the case of the *stationary pattern*, only TSSN ($K = 24$) can complete the missing values with the least amount of error. The reason for this is that both the *upslope pattern* and *downslope pattern* are generally linear with little vibration, however the *stationary pattern* has substantial high-dimensional vibration, making it impossible for the simple model to capture the features.

All of the results demonstrate that the proposed TSSN is capable of extracting high-level and long-term features, making it ideal for a wide range of downstream tasks.

VI. CONCLUSION

In this paper, a token-based SSL network, i.e. TSSN, for TF analysis with a unique pretext task, i.e. MTP, has been proposed. TSSN segments TF data into tokens, and perform token-level operations such as positional encoding. Then, MTP is designed to mask tokens at random, and let TSSN to forecast these tokens according to contextual tokens. Therefore, rather than focusing on point-level correlations and high-frequency details, the MTP enables TSSN to precisely capture the high-level semantics of TF. As a result, the TSSN can attain great performance while retaining a significant ability to generalize.

Fig. 8. 12-point TF completion under different patterns.

Fig. 9. 36-point TF completion under different patterns.

The results demonstrate that TSSNs, especially TSSN ($K = 24$), outperform traditional task-specific models on downstream tasks such as TF classification, prediction and completion.

REFERENCES

- [1] C. Zhang, L. Zhu, J. Ni, C. Huang, and X. Shen, "Verifiable and privacy-preserving traffic flow statistics for advanced traffic management systems," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 10 336–10 347, 2020.

- [2] K. Vijayaraghavan and R. Rajamani, "Novel batteryless wireless sensor for traffic-flow measurement," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 7, pp. 3249–3260, 2010.
- [3] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, 2020.
- [4] E. L. Manibardo, I. Laña, and J. D. Ser, "Deep learning for road traffic forecasting: Does it make a difference?" *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–25, 2021.
- [5] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," *arXiv e-prints*, p. arXiv:2111.06377, Nov. 2021.
- [6] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang, W. Han, M. Huang, Q. Jin, Y. Lan, Y. Liu, Z. Liu, Z. Lu, X. Qiu, R. Song, J. Tang, J.-R. Wen, J. Yuan, W. X. Zhao, and J. Zhu, "Pre-trained models: Past, present and future," *arXiv e-prints*, p. arXiv:2106.07139, Jun. 2021.
- [7] X. Song, Y. Guo, N. Li, and L. Zhang, "Online traffic flow prediction for edge computing-enhanced autonomous and connected vehicles," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 3, pp. 2101–2111, 2021.
- [8] F. Zhao, G.-Q. Zeng, and K.-D. Lu, "Enlstm-wpeo: Short-term traffic flow prediction by ensemble lstm, nnct weight integration, and population extremal optimization," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 101–113, 2020.
- [9] A. Koesdwiady, R. Soua, and F. Karray, "Improving traffic flow prediction with weather information in connected cars: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 9508–9517, 2016.
- [10] C. Zheng, X. Fan, C. Wen, L. Chen, C. Wang, and J. Li, "DeepSTD: Mining spatio-temporal disturbances of multiple context factors for citywide traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3744–3755, 2020.
- [11] X. Feng, X. Ling, H. Zheng, Z. Chen, and Y. Xu, "Adaptive multi-kernel SVM with spatial-temporal correlation for short-term traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 2001–2013, 2019.
- [12] L. Han, K. Zheng, L. Zhao, X. Wang, and X. Shen, "Short-term traffic prediction based on deepcluster in large-scale road networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 12 301–12 313, 2019.
- [13] R. Huang, C. Huang, Y. Liu, G. Dai, and W. Kong, "LSGCN: Long short-term traffic prediction with graph convolutional networks," in *Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 2355–2361, main track. [Online]. Available: <https://doi.org/10.24963/ijcai.2020/326>
- [14] M. G. Cinsdikici and K. Memiş, "Traffic flow condition classification for short sections using single microwave sensor," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, no. 1, p. 148303, 2010.
- [15] K.-H. N. Bui, H. Oh, and H. Yi, "Traffic density classification using sound datasets: An empirical study on traffic flow at asymmetric roads," *IEEE Access*, vol. 8, pp. 125 671–125 679, 2020.
- [16] G. Pastor, "A low-rank tensor model for imputation of missing vehicular traffic volume," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 8934–8938, 2018.
- [17] X. Chen, Y. Cai, Q. Liu, and L. Chen, "Nonconvex l_p -norm regularized sparse self-representation for traffic sensor data recovery," *IEEE Access*, vol. 6, pp. 24 279–24 290, 2018.
- [18] M. Li, L. Wang, J. Yan, H. Zhang, L. Zhang, and L. Zhang, "The study of traffic flow information completion based on gan algorithm," in *2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)*, 2018, pp. 1185–1190.
- [19] L. Han, K. Zheng, L. Zhao, X. Wang, and H. Wen, "Content-aware traffic data completion in ITS based on generative adversarial nets," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 11 950–11 962, 2020.
- [20] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018.
- [21] Y. Yuan and L. Lin, "Self-supervised pretraining of transformers for satellite image time series classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 474–487, 2021.
- [22] Q. Ma, S. Li, W. Zhuang, S. Li, J. Wang, and D. Zeng, "Self-supervised time series clustering with model-based dynamics," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 9, pp. 3942–3955, 2021.
- [23] P. Shi, W. Ye, and Z. Qin, "Self-supervised pre-training for time series classification," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.
- [24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.
- [25] C. Chen, K. Petty, A. Skabardonis, P. Varaiya, and Z. Jia, "Freeway performance measurement system: Mining loop detector data," *Transportation Research Record*, vol. 1748, no. 1, pp. 96–102, 2001. [Online]. Available: <https://doi.org/10.3141/1748-12>
- [26] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.
- [27] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.
- [28] 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 Transactions on Intelligent Transportation Systems*, 2019.
- [29] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Oct. 2014, pp. 1724–1734.