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

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

Abstract—Accurate analysis of traffic flow (TF) data is crucial for the vehicular applications. Conventional deep learning models require task-specific training and are susceptible to high-frequency disturbances, degrading the feature representation capability. To overcome these limitations, this paper proposes a Token-based Self-Supervised Network (TSSN) that can learn TF features in both tokenization and task-agnostic manners. It provides a properly bootstrapped pre-training model for various downstream tasks. In support of the edge computing and vehicular cloud computing, the pooled computational resources facilitate real-time inferences of downstream models. In TSSN, TF data are segmented into tokens. A pretext task, named as Masked Token Prediction (MTP), is then developed to allow TSSN to understand the underlying correlations of TF by predicting randomly masked tokens. By utilizing MTP, TSSN is able to extract the high-level intrinsic semantics of TF, and provide general-purpose token embeddings, leading to improved overall performance and enhanced ability to adapt to different tasks. By substituting the last fully-connected layers with a group of untrained new layers and fine-tuning using small-scale task-specific data, TSSN can be utilized for a variety of downstream tasks in vehicular applications. Simulation results indicate that the TSSN enhances overall performance in comparison to state-of-the-art models.

Index Terms—Masked token prediction (MTP), self-supervised learning, tokenization, traffic flow (TF).

I. INTRODUCTION

Traffic flow (TF) data analytics have shown considerable potentials for mobile applications such as mobile traffic optimization [1], resource allocation [2] and intelligent transportation systems (ITS) [3]. Traffic flow (TF) data represents the number of vehicles crossing a fixed point during a specific time interval [4]. This data is valuable for assessing the transportation situation and facilitating effective traffic route planning or congestion management [5], [6]. Therefore, the precise TF data analysis is crucial to improve the reliability and efficiency of ITS applications [7].

The inherent non-linearity and unpredictability of TF data pose significant challenges in its processing [8]. Meanwhile, TF data might exhibit diverse patterns depending on several circumstances. For example, as a result of the prevalence of roadside structures, TF may show distinct patterns in different blocks of the city [9]. The main goal of this work is to discover the traffic temporal correlations, which involves analyzing the traffic patterns during particular periods of time in different days. One main challenge of TF data analysis is the strong interdependence between the training of a specific model and the specific goal it is intended to achieve. When the purpose is changed or substituted, the trained model may become obsolete, mandating the training of a new model from scratch. Another challenge arises from the necessity of a substantial quality of labeled data to effectively train a deep learning (DL) model. This process demands a considerable period of time and a generous allocation of computational resources [10], [11].

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) [12]. SSL can leverage unlabeled data and an agnostic task to train a model and achieve exceptional performance [13]. Bidirectional encoder representations from transformers (BERT), for example, can learn token correlations in natural language from two pre-training tasks, resulting in the acquisition of general token representations. Nevertheless, despite the inclusion of the Transformer’s attention mechanism, SSL models rely on individual points tend to prioritize short-term correlations and high-frequency disturbances. This may lead to subpar performance in subsequent tasks due to limited generalization ability. Therefore, a token-enabled method is proposed in this paper.
The techniques employed by BERT cannot be directly utilized for processing TF data. TF data consists of sequential numerical values, but natural languages incorporate native words, terms, sentences, punctuation, or other meaningful elements. The TF sequence does not include the concept of linguistic “sentence”. Furthermore, NLP models require training using centralized computational resources provided by cloud computing. This is necessary since the corpus, which is both static and vast, is difficult to learn in scenarios with limited resources. However, when it comes to TF data, it is important to note that they exhibit a high degree of dynamism and are closely associated with spatial placements. A model trained using TF data for one task in one location may not be effectively applied to process TF data for another task or in another location. Thanks to the development of mobile edge computing (MEC) and vehicular cloud computing (VCC), the TF data collected locally in real time can be handled by local models deployed on the edge or the VCC platform. All that is required is a pre-training model capable of effectively extracting the features from TF sequences and generating vector representations that can be utilized for various downstream applications.

Therefore, inspired by BERT, 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 mean concept of this paper is to divide the TF data into tokens and sentences, and subsequently do processing on each token rather than each TF point. A masked tokens prediction (MTP) pretext task is proposed which refers to a task where a random subset of tokens is masked and the pre-training model is required to predict these tokens from contextual tokens in a sentence. This task aims to capture high-level contextual semantics and encode tokens with representations that leverage the fundamental structures of TF data. The proposed tokenization method explores more macroscopic and long-term aspects compared to methods that focus on individual points. Meanwhile, the MTP blurs the boundaries between different tasks, allowing a bootstrapping model to learn the TF semantics for improving the task-specific models’ performance. Ultimately, a general model that comprehends the representation of each token is generated, enabling the accomplishment of specific downstream tasks with exceptional performance by minor fine-tuning. The feasibility and efficiency of the proposed TSSN are demonstrated through a large number of experiments. Our main contributions are summarized as follows, i.e.,

1) A novel tokenization method for TF data is proposed, whereby the long TF sequences are segmented into short tokens. Each token denotes a minimal unit of processing in a pre-training model. It helps improve the interpretation of complex TF data by focusing on long-term characteristics, rather than high-frequency disturbances.

2) A novel pretext task, i.e., masked tokens prediction (MTP), is designed to provide robust surrogate supervision signals for the pre-training of TSSN, based on TF tokenization. Through the MTP, random tokens are masked using random numerical values, and the TSSN is obligated to recover the masked values. The objective is to understand the temporal and token-level correlations of unlabeled TF data, and develop a potent task-agnostic bootstrapping model for various downstream tasks.

3) A multitude of experiments are conducted including three distinct downstream tasks, i.e., TF classification, prediction and data completion, in order to compare TSSN with various approaches. The results clearly indicate the effectiveness of the proposed tokenization method and MTP pretext task.

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, nevertheless, their intricate nature and limited scalability pose challenges for future advancements [14]. DL models have been considered as feature learners in various ways. The main focus of these cutting-edge models is to enhance performance by strengthening the model structure [15], merging multi-modal input [16], or incorporating spatial correlations [17]. Several researches concentrate on the privacy concerns associated with traffic flow analyzing, e.g., [18] and [19]. However, existing literature does not make an effort to extract the features by tokenizing the TF in a manner similar to how words are processed in the NLP field.

A. Traffic Flow Prediction

The TF prediction is essential for a variety of ITS applications. The majority of studies rely on historical data to predict future TF utilizing tensor-based approaches [20], machine learning methods [21] or DL models [22]. The impact of disruptions on TF prediction is investigated by Zheng et al. [23]. They propose two factors: inherent influence factor and disturbance influence factor. These factors are eliminated from TF data to show the underlying patterns. Mallah et al. [24] consider the impact of events to the prediction and propose a deep neural networks (DNN)-based prediction model for TF. Liu et al. [25] propose an conv-sequence TF prediction model that combines spatial and temporal correlations, whereas Han et al. [26] employ an long-short term memory (LSTM) network to make predictions based on the representation learning of shape-based features. Liu et al. [27] utilize a graph processing technique to estimate the city-wide traffic, while Zhang et al. [28] propose s a graph neural networks (GNN)-based model that can effectively manage dynamic scenario changes during TF forecasting. Both of these studies concentrate on short-term predictions (less than an hour). For both short-term and long-term TF predictions, Huang et al. [29] present a graph convolutional networks (GCN)-based model (from 5 min to 60 min). By introducing spatial features, the model can attain a high level of performance.
B. Traffic Flow Classification

Additional groups arise from the TF data when different contextual circumstances, such as weather, accidents, weekends, or vacations, are considered. Cinsdikici et al. [30] offer a two-phase approach to accurately measure fluctuations in TF density and classify TF patterns at each specific moment in time. The TF can be categorized into three distinct types: free flow, the dense flow and the congested flow. Bui et al. [31] investigate the potential of utilizing traffic sound data for the purpose of categorizing 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, specifically in the morning or evening.

C. Traffic Flow Data Completion

Tensor completion is used to model the TF data using high-dimensional tensors and generates low-rank representations using high-order decomposition. The completion problem is then solved using norm regularization [32], [33]. However, due to their considerable complexity, these strategies are difficult to be put into practice. The neural network-based TF data completion is highly efficient and capable of accurately filling in extended periods of missing data. Li et al. [34] investigate the potential application of generative adversarial networks (GAN) in filling in missing TF data using a graph-based approach. Meanwhile, Wang et al. [35] propose a hybrid model for more effective TF imputation. Han et al. [36] offer a GAN-based data completion method that combines tensor modeling to achieve long-term TF data completion. The main concept is to use low-dimensional representations to recover the most appropriate TF data sequences. The results indicate that the performance is satisfactory even completing data for a duration of one week.

D. Self-Supervised Learning on Time Series

SSL is becoming more popular as a technique for extracting features from datasets that are difficult to label. The power of SSL in NLP is particularly remarkable due to the implementation of Transformer structure. One highly successful models is Google’s BERT, which leverages Transformer encoders to learn knowledge about sentence representations [37]. In BERT, tokens undergo random masking, and two pretext tasks, namely masked language modeling (MLM) and next sentence prediction (NSP), are devised during the pre-training stage. Yuan et al. [38], inspired by BERT, develop a self-supervised pre-training model that gains general knowledge from large-scale unlabeled satellite image time series. This knowledge is then utilized for classification tasks with scarce-label data. Ma et al. [39] present a time series clustering approach in which the pseudo-class labels are generated through the k-means algorithm. As a result, clustering can be accomplished without a huge amount of labeled data. Shi et al. [40] devise two new pre-training tasks for acoustic time series classification. The proposed model can be utilized to increase performance on tiny labeled datasets.

E. Pre-Training Models on TF Data

Inspired by NLP models, pre-training and fine-tuning paradigm is adopted to train DNNs for TF data. Jin et al. [41] completely use the structure of BERT and pre-training method for traffic forecasting. Shao et al. [42] propose a pre-training method for the application of forecasting traffic time series. They consider both the spatial and temporal correlations with masking mechanism, but they did not provide a thorough explanation of the specific masking methods used. Similarly, Gao et al. [43] propose a masked pre-training for traffic forecasting applications by masked auto-encoder. The masking method is not further elaborated upon. Furthermore, each of these two researches only focuses on applications related to traffic forecasting. Another Transformer pre-training model for traffic forecasting is proposed by Zhang et al. [44], where they use multiple time step embeddings and learnable positional embeddings. The masks are randomly applied to the input sequences in the embedding domain, rather than the time domain. Qu et al. [45] design an intricate pre-training model comprising of four pre-training tasks. The loss function is the linear combinations of all tasks. In terms of the masking method, the authors use the structure of masked auto-encoder, without providing any additional clarification.

Overall, BERT has achieved tremendous success in the domain of NLP, it cannot be directly applied for the processing of TF data due to the fact that TF data does not contain the linguistic elements of words and punctuation. Additionally, the numerical values themselves are indicative of the specific features of the TF data. Current researches on TF data processing typically focus on particular tasks. For instance, the model proposed in [29] is limited to making predictions on TF data exclusively, whereas [36] is specifically designed for TF data completion. The state-of-the-art models that using pre-training sometimes fail to adequately consider masking strategies. There have been limited efforts to develop task-agnostic and tokenization methods that can be applied in various areas of ITS applications. Therefore, employing token-based processing and pre-training techniques, this work principally presents an SSL model that may effectively facilitate the bootstrapping for various downstream tasks.

III. TRAFFIC FLOW DATA TOKENIZATION

Let $X$ denote the dataset which contains traffic flow data of several days on multiple road segments. Let $N$ denote the total number of points for one day. All points in one day are treated as a sentence. Let $x_i \in X$, $i = 1, \ldots, N$ 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 $t_j$, $j \in [1, M]$ denote the $j^{th}$ token in one sentence, then we have $t_j := x_{(j-1)K+1}, x_{(j-1)K+2}, \ldots, x_{jK}$. The rationale is to treat consecutive $K$ points as a unit and learn the features or patterns that it possesses. The value of $K$ can be arbitrarily determined, theoretically, and different tokens can have different number of points. However, only constant value of $K$ is considered in order to evaluate the efficacy of tokenization method. Additionally, $K$
needs to satisfy that $M = N/K \in \mathbb{Z}$, otherwise, there will be incomplete sentence.

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

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

A part of tokens is masked with random numbers to let the model learn the representations in self-supervised manner. Let $\mathcal{U}(a, b)$ denotes the uniform distribution whose probability density function is $f(x) = \frac{1}{b-a}, a \leq x \leq b$. All points in masked tokens are added by positive or negative random values, i.e.,

$$\xi_j = \begin{cases} x_{(j-1)K+1} + \delta, & \text{if } \xi_j < \alpha, \\
 x_{(j-1)K+1}, & \text{if } \xi_j \geq \alpha.
\end{cases}$$  \hfill (2)

where $\xi_j$ is the $j$\textsuperscript{th} random number that follows $\mathcal{U}(0, 1)$, and $\alpha$ stands for the masking ratio. The masking value, denoted as $\delta$, is obtained by sampling from $\mathcal{U}(-b, b), b > 0$. That is to say, there is a 50 \% probability of introducing a positive noise, and a 50 \% probability of adding a negative noise. Therefore, by compelling the model to recover the original values from noised inputs and context, it can enhance the capacity of generalization. Let $\mathbf{m}$ record whether the element of a specific position in input sequence is masked. Then, it can be defined as follows,

$$\mathbf{m}_{(j-1)K:jK} = \begin{cases} \{1, 1, \ldots, 1\}, & \xi_j < \alpha, \\
 \{0, 0, \ldots, 0\}, & \xi_j \geq \alpha.
\end{cases}$$  \hfill (3)

IV. PRE-TRAINING MODEL WITH MASKED TOKEN PREDICTION

A. Model Structure

Assume each input is denoted as $x \in \mathbb{R}^N$. 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 $t$ is embedded to a $D$-dimensional vector as $\mathbf{t}$ 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,

$$\bar{\mathbf{X}} = \mathbf{XW}^{\text{Embedding}},$$  \hfill (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 [46], i.e.,

$$p_{j,d} = \begin{cases} \sin \left( \frac{j \times 10000^d}{D} \right), & \text{if } d \mod 2 = 0, \\
 \cos \left( \frac{j \times 10000^d}{D} \right), & \text{if } d \mod 2 = 1,
\end{cases}$$  \hfill (5)

where $j \in [1, M]$ and $d \in [1, D]$. The base value 10000 is chosen according to [46]. The $p_{j,d}$ is add to the $\bar{\mathbf{X}}$ element-wisely, i.e.,

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

Each point in a token must add identical positional encodings (PEs), which correspond to the PEs of the current token’s position. As a result, PEs incorporate token-level positional information rather than point-level positional information. This ensures that the model considers the points within a single token as a cohesive unit, allowing the model to prioritize high-level semantic features, but not being affected by the high-frequency disturbances between points.

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 calculated by the multiply of $\bar{\mathbf{X}}$ and a learnable weight matrix. Then, the attention is computed, after which the multi-head attentions can be derived by the concatenation of $L$ heads with linear projections by a fully-connected layer. Denote the outputs of encoder as $\mathbf{A}$, its dimension is equivalent to $\bar{\mathbf{X}}$. The detailed equations can be found in [46].

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 [46]. Let $\bar{\mathbf{X}} \leftarrow \mathbb{I}(\bar{\mathbf{X}} + \mathbf{A})$, then the representations can be calculated as follow,

$$\bar{\mathbf{Y}} = g(\bar{\mathbf{XW}}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2,$$  \hfill (7)

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 $\mathbf{W}_2 \in \mathbb{R}^{D^{\text{hid}} \times D}$, $\mathbf{b}_2 \in \mathbb{R}^{D \times 1}$.

Note that only Transformer encoder is used in this model, and the transformer encoders’ outputs $\bar{\mathbf{Y}}$ are the encoded representations of each token in a sentence. This layer is named as the 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.,

$$\mathbf{y} = (\bar{\mathbf{YW}}_3 + \mathbf{b}_3)^{\text{flatten}},$$  \hfill (8)

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

B. MTP Pretext Task

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.,

$$L^\text{pre} = \mathbb{E}_{t \sim \mathbb{X}} \left\{ \sum_{j=1}^M \left[ \mathbf{m}_j \odot (\mathbf{y}_{(j-1)K:jK} - t_j) \right]^2 \right\},$$  \hfill (9)

where $\theta$ stands for all trainable parameters of the model. Only masked tokens are considered in the loss calculation, according
to eq (9). Thus, TSSN can learn more abstract and long-term token-level features because the masked tokens are randomly chosen and changed.

C. Fine-Tuning

To fine-tune TSSN for various downstream tasks, the TSSN’s final fully-connected layer is substituted by new layers with untrained parameters. For the fine-tuning tasks, the pre-training model exclusively generates the embedding representations \( Y \) of each token in a sentence. With task-specific labeled data, a bit more training is added. Furthermore, the loss function is determined by the downstream task’s purpose. During the fine-tuning process, only the parameters of posterior layers are modified. The parameters in pre-training model remain unchanged.

To evaluate the effectiveness 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 \( Y \), i.e.,

\[
y^C = \mathbb{S} \left\{ g \left( \sum_{i} \left( Y W^C_1 + b^C_1 \right) W^C_2 \right) \right\},
\]

where \( W^C_1 \in \mathbb{R}^{D \times D^{C-hid}}, b^C_1 \in \mathbb{R}^{D^{C-hid} \times 1}, \) and \( W^C_2 \in \mathbb{R}^{D^{C-hid} \times S} \). The final output \( y^C \) becomes a \( S \)-dimensional vector, each element \( y^C_s \) of which represents the probability of being classified as the \( s \)th category.

The labeled dataset for classification task is denoted as \( x^C = \{(x, \rho)\}, \) where \( \rho = [\rho_1, \rho_2, \ldots, \rho_S] \), \( \rho_1, \rho_2, \ldots, \rho_S \in \{0, 1\} \) and \( \sum_{s=1}^{S} \rho_s = 1 \). \( \rho \) stands for the one-hot encoding of categorical labels where \( \rho_s = 1 \) means \( x \) 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.,

\[
J^C = -\mathbb{E}_{x \sim X} \left\{ \sum_{s=1}^{S} \rho_s \log y^C_s \right\}.
\]

2) Traffic Flow Prediction: To make precise predictions on the future horizon of TF based on existing data, TSSN is reconstructed by connecting two dense layers after the representation layer, i.e., for \( P \)-horizon prediction task, the output \( y^P \) is given as follow,

\[
y^P = g \left( \sum_{i} \left( Y W^P_1 + b^P_1 \right) W^P_2 + b^P_2 \right),
\]

where \( W^P_1 \in \mathbb{R}^{D \times D^{P-hid}}, b^P_1 \in \mathbb{R}^{D^{P-hid} \times 1}, \) and \( W^P_2 \in \mathbb{R}^{D^{P-hid} \times P}, b^P_2 \in \mathbb{R}^{P \times 1} \), 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.,

\[
J^P = -\mathbb{E}_{x \sim X} \left\{ (y^P - x)^2 \right\},
\]

where \( x^P \) 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 selected in a sequential manner using randomization. Given that the total missing number is \( C \), the task’s fine-tuning model can be expressed as follows,

\[
y^{CM} = g \left( \sum_{i} \left( Y W^{CM}_1 + b^{CM}_1 \right) W^{CM}_2 + b^{CM}_2 \right),
\]

where \( W^{CM}_1 \in \mathbb{R}^{D \times D^{CM-hid}}, b^{CM}_1 \in \mathbb{R}^{D^{CM-hid} \times 1}, \) and \( W^{CM}_2 \in \mathbb{R}^{D^{CM-hid} \times C}. \) 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 \( \beta \) is used to replace the missing data. Assume that the missing places \( m^{CM}_i \) are represented by a binary vector of \( N \) elements, each of which is specified as follow,

\[
m_i = \begin{cases} 1, & \text{If } x_i \text{ is missing}, \\ 0, & \text{Otherwise}. \end{cases}
\]

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

\[
J^{CM} = \mathbb{E}_{x \sim X} \left\{ \| m^{CM} \odot (y^{CM} - x) \|^2 \right\}.
\]

In contrast to the pre-training tasks, missing data is substituted with a fixed value instead of a random number. Furthermore, unlike the prediction task where only the numbers at expected positions are outputted, the output \( y^{CM} \) is an entire 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 minutes), \( K = 2 \) (ten minutes), \( K = 6 \) (half an hour), \( K = 12 \) (one hour) and \( K = 24 \) (two hours). \( D = 64 \) is the embedding dimension, which means that each token is represented by a 64-dimension vector. The source sentence embedding is fed into three repeated transformer encoders. The number of simultaneous attention 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) [47]. 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 hour and \( N = 12 \times 24 = 288 \) points per day. The data in District 3, 4, 5 and 7 from Aug. 31, 2018 to Feb. 1, 2019 are mixed up to train and test our models. 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 (1) before training. The pre-training data are randomly masked according to (2), where the mask rate \( \alpha = 15\% \) and the boundary of masking value \( b = 0.5 \). The value range of normalized TF data is between [0, 1]. Thus, the boundary of uniform distribution \( b = 0.5 \) ensures that the masked token can be the original token (with little deviation, \( \delta \approx 0 \)), the original token with slight noises (0 \( \leq |\delta| \ll b \)) and a completely random token (0 \( \leq \delta \leq b \)).

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

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
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 [48] is used throughout the whole TSSN architecture. All of the tests in this paper were run on NVIDIA Tesla graphics processing units (GPUs) using compute unified device architecture (CUDA) 10 and PyTorch.1 Such an experiment setting considers the possibility of deploying the model on the edge or VCC. The whole program is open sourced on Github.2

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

a) Weekday-Weekend Classification: One of the categorical downstream tasks of TSSN is the weekday-weekend classification task. The objective is to classify the day as either a weekday or a weekend day according to the distinct properties and patterns of TF data. Understanding the variation characteristics in TF data under various circumstances, such as day types, considerably enhances our comprehension of the operational aspects of urban transportation. This task makes use of a dataset known as Seattle Inductive Loop Detector Dataset [49], [50]. The data are collected in the same manner as PeMS. Meanwhile, each day is categorized as either a weekday or a weekend, with the label $\rho = [0, 1]$ denoting weekday and $\rho = [1, 0]$ representing weekend. The hidden dimension is $D_{C-hid} = 64$. A 100 epochs of fine-tuning are conducted, with each epoch being followed by a validation phase. Two baselines are introduced, namely three-layer fully connected (FC) network and gated recurrent unit (GRU) network [51], for the purpose of comparing several metrics such as average classification accuracy, precision, recall, F1 score and Kappa coefficient.

b) $P$-Horizon Prediction: Downstream prediction tasks encompass both short-term and long-term predictions. For short-term predictions, the values of $P$ are set to be 1, 6, 12 which correspond to predictions with horizons of 5 min, 30 min and 1 hr, respectively. On the other hand, for long-term predictions, $P$ is set to 36 and 72, representing horizons of 3 hrs and 6 hrs, respectively. The hidden size $D_{P-hid}$ is set to 64, and the fine-tuning processes terminates exclusively when the validation losses have reached a stable state. Comparative approaches include a bidirectional long-short term memory (BiLSTM) network [49] and a Transformer network [46]. Besides, several results from published articles are also introduced to provide a comparison with the proposed method. The metrics consist of mean absolute error (MAE), root mean square error (RMSE), mean absolute percentage error (MAPE) and $R^2$.2

c) $C$-Point Data Completion: Under varying values of $K$, two tasks, namely 12-point (1 hr) and 36-point (3 hrs) data completion, are tested. The missing values are replaced with a value of $\beta = 0$, and the hidden layer’s size is set to $D_{Cm-hid} = 64$. The baselines consist of a three-layer FC network and a Transformer network [46]. In addition, a comparison is made between TSSN and a GAN-based model proposed in [36]. The metrics remain the same as in prediction tasks.

B. Results and Analysis

1) Convergence: The convergence of all pre-training models is demonstrated in Fig. 2(a), where they reach a low loss value and maintain stability. Evidently, tokens with a smaller number of points ($K$ value) exhibit a correspondingly lower loss value. The reason for this is that a larger $K$ value increases the complexity of learning, but it enables the capture of long-term temporal correlations. The convergences of different downstream tasks are shown in Fig. 4. Following the initial 20 epochs of training, the validation accuracy values of all TSSNs exceed 95%, and upon convergence, they reach an accuracy above 98% for the classification task. The FC model exhibits longer convergence time compared to all the TSSN models, but the ultimate result is deemed satisfactory. The GRU exhibits a gradually convergence towards a low loss value, rendering it unsuitable for the classification of weekdays and weekends. Fig. 2(c) and (d) show that for both prediction and completion tasks, each model can converge to a low loss value, but with varying convergence speeds.

2) Weekday-Weekend Classification: Table III displays the classification task’s results, where TSSN($K = 2$) achieves the highest accuracy. TSSNs exhibit superior accuracy compared to FC and GRU network, implying that TSSN has the potential to greatly improve performance in classification tasks. The Kappa coefficients indicates the validity and reliability of all the TSSNs. The classification consistency of FC network can only be regarded as adequate and GRU network can only be considered moderate.

3) Traffic Flow Prediction: According to the literature, $P \leq 12$ (1 hr) is commonly used to imply short-term prediction, whereas $P > 12$ is used to signify long-term prediction for TF data. Furthermore, a MAPE score of 20% is typically deemed as the threshold between model being considered available and unavailable.

a) Short-Term Prediction: As shown in Table IV, BiLSTM achieves the highest performance for the 1-horizon (5 min)
Fig. 2. Illustration of the convergence of different tasks.

Fig. 3. Illustration of continuous seven-day prediction results of TSSN ($K = 12$) under different horizons.

Fig. 4. Illustration of the prediction task performance of different models under different horizons.

In terms of the MAPE score, the highest performance of TSSN ($K = 12$) falls behind BiLSTM by around 61.22%. The reason for this is because BiLSTM models excel at capturing short-term features, whereas TSSNs are better at capturing intermediate- and long-term characteristics. Since the MAPE score is approaching 20%, the Transformer network, as well as the TSSN ($K = 1$), which simply considers individual points and ignores the intrinsic correlations between consecutive points, are unavailable for this task. TSSNs outperform the baselines in both 6-horizon (30 min) and 12-horizon (1 hr) prediction tasks, as evidenced by the MAPE score. The reason for this advantage is that TSSN has the capability to understand the characteristics of TF tokens, hence enhancing its ability to extract internal patterns of TF.

b) Long-Term Prediction: The MAPEs of the TSSNs from $K = 2$ to $K = 24$ are below 20% for the 36-hour (3 hrs) prediction task, as illustrated in Table V. TSSN ($K = 12$) has performance among all of them. The TSSN ($K = 24$) slightly falls behind TSSN ($K = 12$) by a margin of 4.7%. The MAPE score of TSSN ($K = 1$) exceeds 20% by 9.7%, indicating that it is unavailable. This result highlights the effectiveness of using a token-based pre-training model for long-term prediction tasks. The BiLSTM and Transformer models are completely unavailable for this task.

The prediction task with a 72-hour (6 hrs) can be significantly more demanding than the 36-hour task. The TSSN ($K = 12$) has the highest performance with a MAPE score that is just 0.5% above the threshold. Meanwhile, TSSN ($K = 24$) lags behind TSSN ($K = 12$) by 0.8%. The other models, particularly the two baselines, are completely inaccessible for this task.

c) Seven-Day Continuous Prediction: Fig. 3 depicts the continuous seven-day prediction results of TSSN ($K = 12$) for various $P$ values. The model consistently generates predictions for $P$ values and constructs for a total duration of 7 days. As the value of $P$ increases, the predicted results gradually deviate from the genuine values. Even for a prediction task with a complexity of $P = 36$, the model is capable of capturing the overall patterns but fails to accurately match the rapid fluctuations. Appendix
contains additional 7-day prediction results from various models to showcase the accuracy of predictions.

d) Metrics on Different Horizons: In Fig. 4(a), we compare the TSSN ($K = 12$) against TSSN ($K = 1$), as well as the BiLSTM and Transformer across various lengths of predicted horizons. The evaluation is conducted using the MAPE metric. The BiLSTM performs best at the beginning, specifically when
When comparing the Transformer model to the TSSN ($K = 1$), which applies a pre-training processing without tokenization, the TSSN's results show underwhelming scores for all forecasting horizons. This implies that the Transformer structure may not be ideally for the TF prediction tasks, and that relying solely on pre-training without tokenization cannot improve the efficacy of prediction task. Except for when $P = 1$, the TSSNs ($K = 12$) outperform all other cases. Therefore, it is evident that the proposed approach can substantially improve prediction capability. The similar trends can be revealed over other metrics in Fig. 4.

**e) Comparisons to Related Methods:** We compare the short-term prediction performance of our most effective TSSN ($K = 12$) model to several cutting-edge models, i.e., STEP [42], STD-MAE [43], AT-Conv-LSTM [52], STLGru [53], PDFormer [54]. The STEP model incorporates spatial and temporal correlations and proposes a GNN based pre-training model. STD-MAE leverages the structure of masked auto-encoder while additionally considering the spatial and temporal correlations. The AT-Conv-LSTM model consists of attention-enabled convolutional layers and LSTM layers, while the STLGru model takes the spatial features into account, and is composed of GCN and GRU layers. PDFormer model utilizes spatial and temporal embeds along with a delay-aware feature transformation. Table VI demonstrates that, with the exception of STD-MAE, which achieves superior prediction accuracy on 12 horizons, all other related methods fall slightly below our TSSN ($K = 12$). This demonstrates the effectiveness of both tokenization in TF data and MTP pretext task. While the primary objective is to develop a model for the bootstrap of multiple tasks rather than achieving optimal performance for a single task, the TSSN is nevertheless capable of outperforming certain specialized models.

**4) Traffic Flow Data Completion:** According to Table VII, the TSSN ($K = 24$) has the highest overall completion performance compared to the other models. The MAPEs of all TSSN models are less than $20\%$ for 12-point completion. However, for 36-point completion task, only TSSN ($K = 12$) and TSSN ($K = 24$) are available, as their MAPEs are below the $20\%$ criterion.

For better illustrating the efficacy of TSSN, three typical patterns are identified for both 12-point and 36-point TF completion tasks. These patterns include the upslope pattern, the stationary pattern and the downslope pattern. Each pattern represents the TF's ascending, stable, and descending trends, respectively. A random day is chosen to evaluate the performance of different models. The data presented in Fig. 5 indicates that TSSN ($K = 24$) is effective in accurately filling in the absent

### Table V: Performance of Long-Term Prediction

| Task | 36-hour (3 hrs) prediction |
|------|----------------------------|
| $K = 1$ | 9.0532 | 14.5817 | 21.94% | 0.9091 |
| $K = 2$ | 7.6587 | 12.4783 | 18.39% | 0.9334 |
| $K = 6$ | 8.4476 | 15.8594 | 19.20% | 0.8961 |
| $K = 12$ | 6.0974 | 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 |

The bold values are the best values among all the results.

### Table VI: Comparisons between TSSN and Related Methods Regarding to Prediction MAPE

| Model | Task | 1-horizon | 6-horizon | 12-horizon |
|-------|------|-----------|-----------|------------|
| STEP [42] | N/A | N/A | 10.75% |
| STD-MAE [43] | N/A | N/A | 11.13% |
| AT-Conv-LSTM [52] | 10.1% | 11.4% | 12.3% |
| STLGru [53] | N/A | 11.67% | 12.6225% |
| PDFormer [54] | N/A | N/A | 11.01% |

* The results of STLGru, STD-MAE, STEP and PDFormer are the average MAPEs over four datasets to match our experimental settings.

### Table VII: Performance of TF Data Completion

| Task | 12-point (1 hr) 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% | 0.9920 |
| $K = 24$ | 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 (3 hrs) completion |
|------|-----------------------------|
| $K = 1$ | 12.1169 | 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 |
| $K = 12$ | 10.3509 | 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 |

The bold values are the best values among all the results.
values in the upslope pattern of 12-point completion. Furthermore, it outperforms all four methods in both stationary pattern and downslope pattern, while not achieving optimal matching of missing values. This finding highlights the capability of TSSN to enhance accuracy in the task of TF completion.

To provide a clear illustration, the results of 36-point completion are sampled at intervals of 4 points, resulting in a total of only 10 points on the curves. As shown in Fig. 6, all models are capable of accurately filling in the missing values, with only a slight bias observed in both the upslope pattern and downslope pattern. In terms 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 exhibit a predominantly linear trend with little vibration. In contrast, the stationary pattern has substantial high-dimensional vibration, making it unfeasible for the simple model to accurately represent its characteristics.

Aside from the comparisons among different values of \(K\) and two baseline models, we have also compared our model against a related method proposed by [36]. This model combines GAN with tensor-based completion approaches to achieve various TF completion tasks. The MAPE score for 36-point completion slightly exceeds 15% in [36]. Our TSSN \(K = 24\) can achieve a comparable result (14.78%) while maintaining generality for additional types of downstream tasks.

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

VI. CONCLUSION

This paper proposes a token-based SSL network named TSSN, which is designed for TF analysis. The network incorporates a unique pretext task, i.e., MTP. The TSSN segments TF data into tokens and perform token-level operations, such as positional encoding. Next, MTP is designed to randomly mask tokens, letting TSSN to forecast these tokens according to contextual information. Therefore, instead of 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 achieve exceptional performance while retaining a significant ability for generalization. The pre-trained model can be deployed on MEC or VCC platforms, enabling them to analyze real-time local TF data for different applications. The results demonstrate that TSSNs outperform conventional task-specific models in terms of performance on downstream tasks such as TF classification, prediction and completion.

On the basis of this paper, our future work will focus on integrating the temporal correlations with spatial correlations of TF data. We aim to design a tokenization-based pre-training model with MTP task and GNN. Thus, both the temporal and spatial information can be effectively employed to achieve optimal performance of TF data analytics.

REFERENCES

[1] J. Prados-Garzon, T. Taleb, and M. Bagaa, “Optimization of flow allocation in asynchronous deterministic 5G transport networks by leveraging data analytics,” IEEE Trans. Mobile Comput., vol. 22, no. 3, pp. 1672–1687, Mar. 2023.
[2] J. Zhou et al., “Joint energy-efficiency communication optimization and perimeter traffic flow control for multi-region LTE-V2V networks,” IEEE Trans. Mobile Comput., vol. 23, no. 5, pp. 5009–5026, May 2024.
[3] J. Han, G. H. Lee, S. Park, J. Lee, and J. K. Choi, “A multivariate-time-series-prediction-based adaptive data transmission period control algorithm for IoT networks,” IEEE Internet Things J., vol. 9, no. 1, pp. 419–436, Jan. 2022.
[4] Y.-e. Sun, H. Huang, S. Chen, H. Xu, K. Han, and Y. Zhou, “Persistent traffic measurement through vehicle-to-infrastructure communications in cyber-physical road systems,” IEEE Trans. Mobile Comput., vol. 18, no. 7, pp. 1616–1630, Jul. 2019.
[5] Y. Wang, T. Xu, X. Niu, C. Tan, E. Chen, and H. Xiong, “STMARL: A spatio-temporal multi-agent reinforcement learning approach for cooperative traffic light control,” IEEE Trans. Mobile Comput., vol. 21, no. 6, pp. 2228–2242, Jun. 2022.
[6] J. Pan, I. S. Popa, and C. Borcea, “DIVERT: A distributed vehicular traffic re-routing system for congestion avoidance,” IEEE Trans. Mobile Comput., vol. 16, no. 1, pp. 58–72, Jan. 2017.
[7] Y. Jiao, K. Yang, D. Song, and D. Tao, “TimeautoAD: Autonomous anomaly detection with self-supervised contrastive loss for multivariate time series,” IEEE Trans. Netw. Sci. Eng., vol. 9, no. 3, pp. 1604–1619, May-Jun. 2022.
[8] K. Vijayaraghavan and R. Rajamani, “Novel batteryless wireless sensor for traffic-flow measurement,” IEEE Trans. Veh. Technol., vol. 59, no. 7, pp. 3249–3260, Sep. 2010.
[9] T. Qi, G. Li, L. Chen, and Y. Xue, “ADGCN: An asynchronous dilation graph convolutional network for traffic flow prediction,” IEEE Internet Things J., vol. 9, no. 5, pp. 4001–4014, Mar. 2022.
[10] L. Nie, Z. Ning, X. Wang, X. Hu, J. Cheng, and Y. Li, “Data-driven intrusion detection for intelligent Internet of Vehicles: A deep convolutional neural network-based method,” IEEE Trans. Netw. Sci. Eng., vol. 7, no. 4, pp. 2219–2230, Oct.–Dec. 2020.
Yunxing Geng received the BS degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2023. He is currently an assistant engineer with China Satellite Network Group Company, Ltd. His research focuses on deep learning.

Lingyi Han received the bachelor’s, master’s, and doctoral degrees from the Beijing University of Posts and Telecommunications, Beijing, China, in 2013, 2016, and 2020, respectively. She is currently an engineer with China Aero Geophysical Survey & Remote Sensing Center for Natural Resources. Her research interests include data mining and artificial intelligence in remote sensing.

Haojun Yang (Member, IEEE) received the BS degree in communication engineering and the PhD degree in information and communication engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2014 and 2020, respectively. He is currently a postdoctoral fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada. His research interests include ultra-reliable and low-latency communications, radio resource management and vehicular networks.

Kan Zheng (Fellow, IEEE) received the BS, MS, and PhD degrees from Beijing University of Posts and Telecommunications, China, in 1996, 2000, and 2005, respectively. He is currently a full professor with Ningbo University, Ningbo, Zhejiang, China. He has rich experience in research and standardization of new emerging technologies. He has authored more than 200 journal articles and conference papers in the field of wireless communications, vehicular networks, IoT, and security. He holds editorial board positions with several journals and also served in the organizing/TPC committees for conferences.

Xianbin Wang (Fellow, IEEE) received the PhD degree in electrical and computer engineering from the National University of Singapore, in 2001. He is currently a professor and a tier-1 Canada research chair in 5G and Wireless IoT Communications with Western University, Canada. Prior to joining Western University, he was with the Communications Research Centre Canada as a Research Scientist/Senior Research Scientist from 2002 to 2007. From 2001 to 2002, he was a System Designer at STMicroelectronics. He has more than 600 highly cited journals and conference papers, in addition to more than 30 granted and pending patents and several standard contributions. His current research interests include 5G/6G technologies, Internet of Things, machine learning, communications security, and intelligent communications. Dr. Wang is a fellow of the Canadian Academy of Engineering and a fellow of the Engineering Institute of Canada. He was the recipient of many prestigious awards and recognitions, including the IEEE Canada R. A. Fessenden Award, Canada Research Chair, Engineering Research Excellence Award at Western University, Canadian Federal Government Public Service Award, Ontario Early Researcher Award, and nine Best Paper Awards. He was involved in many IEEE conferences, including GLOBECOM, ICC, VTC, PIMRC, WCNC, CCECE, and CWIT, in different roles, such as General Chair, TPC Chair, Symposium Chair, Tutorial Instructor, Track Chair, Session Chair, and Keynote Speaker. He is/was the Editor-in-Chief, Associate Editor-in-Chief, and editor/associate editor for over ten journals. He was the Chair of the IEEE ComSoc Signal Processing and Computing for Communications (SPCC) Technical Committee and is currently serving as the Central Area Chair of IEEE Canada.