Spatial-temporal Transformer Network with Self-supervised Learning for Traffic Flow Prediction

Zhangzhi Peng*, Xiaohui Huang

East China Jiaotong University

Abstract
Traffic flow prediction plays a critical role in improving the quality, security, and efficiency of Intelligent Transportation Systems (ITS). Accurate prediction requires modeling spatial and temporal characteristics simultaneously. Existing works usually extract the spatial features by CNN-based modules and temporal features by RNN-based modules. However, the CNN-based modules are locally biased, performing poorly in global spatial dependencies; and the RNN-based modules concentrate on learning the high-level temporal dynamics (e.g., periodicity), and fail to consider the numerical closeness between future data and historical observations as a strong prior knowledge for the prediction. To alleviate these limitations, we propose a Spatial-temporal Transformer Network with Self-supervised Learning (ST-TSNet). ST-TSNet uses a Pre-Conv Block and vision transformer to learn the spatial dependencies in both local and global contexts. Furthermore, a skip connection from the input of historical records to the output prediction is introduced to utilize similar patterns to improve the prediction results. Finally, a self-supervised strategy called stochastic augmentation is proposed to explore spatial-temporal representations from massive traffic data to benefit the prediction task. Experiments on two datasets, TaxiBJ and TaxiNYC, demonstrate the effectiveness of ST-TSNet. The codes is available at https://github.com/pengzhangzhi/spatial-temporal-transformer.

1. Introduction

Traffic flow prediction is a build block in Intelligent Transportation Systems (ITS), which is essential for providing high-quality traffic service. An accurate prediction of future traffic flow data depends on modeling the spatial-temporal information from the previous observations. This problem can be considered from the spatial and temporal perspectives. From the spatial perspective, learning the local spatial correlations is essential since traffic volume is most influenced by its nearest neighbors. However, in real-world scenarios, two distant regions may be strongly correlated in their traffic distributions as they feature the similar functionality (e.g., transportation hub). Most of existing works [1, 2, 3] adopt the convolutional layers as their backbone to extract the spatial features, which may introduce short-range bias due to their small receptive field. These methods perform well in extracting local context while hindering in global dependencies. Recently, Vision transformer (ViT) [4] has shown impressive performance in computer vision, due to its innate power at extracting non-local features. We are motivated to apply ViT to learn the long-range spatial dependencies.

From the temporal perspective, many works have been proposed to extract complex temporal patterns, e.g., daily and weekly periodicity [1, 2]. However, we argue that a simple temporal characteristic: temporal similarity is overlooked. Traffic flow data are generally smooth with few abrupt changes, showing many similarities in adjacent frames. As depicted in the time series of Fig.1, the ratio of current traffic flow to the previous one (shown in blue line) floats up and down within a fixed ratio of 1 as the traffic flow (shown in orange line) periodically evolves. This means that adjacent traffic flow snapshots have a close value and exhibit similar distribution. Thus, an intuitive idea is to use historical observations as the base prediction for future data. Such motivation provides a prior knowledge that forces the model to predict the future data partially based on the original historical records instead of completely depending on the extracted temporal patterns. However, such similarity is overlooked in existing methods [5, 2], as they process the historical data for high-order temporal characteristics (e.g., periodicity), distorting the numerical similarity.

With the rapid growth of traffic sensors deployed, a massive amount of traffic flow data is collected but not fully utilized. Similarly, in the field of natural language processing (NLP), TB-level unlabel corpus are collected but relatively fewer label data is available for various language tasks. The gap , however, in NLP is successfully alleviated by self-supervised learning [6], where unlabel data are utilized to learn language representations and then transferred to facilitate downstream tasks. While in the field of traffic flow prediction, current training algorithms are supervised learning, where the historical records are regarded as input and the traffic data in the next timestamp is served as label. No effective unsupervised learning algorithms are proposed to learn spatial-temporal representations to facilitate the traffic flow prediction task.

Driven by these analyses, we propose a novel framework called Spatial-temporal Transformer Network with

STRL'22: First International Workshop on Spatio-Temporal Reasoning and Learning, July 24, 2022, Vienna, Austria
*Corresponding author.
pengzhangzhics@gmail.com (Z. Peng); hxh016@gmail.com (X. Huang)

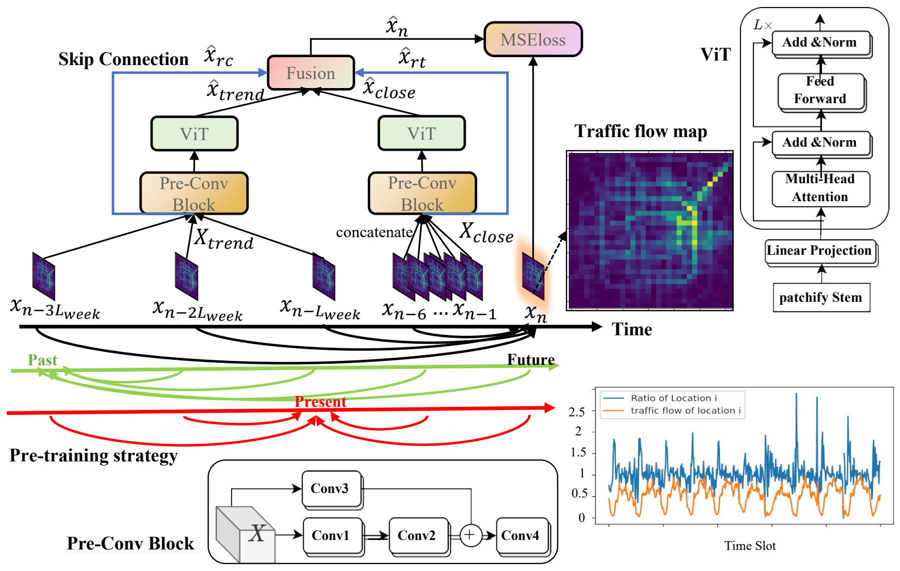

Figure 1: The overall architecture of Spatial-temporal Transformer Network with Self-supervised Learning (ST-TSNet). The three time axes illustrate our pre-training strategy. The time series at the bottom shows the periodicity of traffic flow data; the blue line denotes the ratio, and the orange line denotes the normalized traffic flow observations. The figure reveals that as the traffic flow periodically changes, the ratio floats up and down from a fixed value 1.

Self-supervised Learning (ST-TSNet). ST-TSNet consists of a Pre-Conv Block and ViT for learning spatial correlations in both local and global contexts. In addition, we directly connect the historical data to the output to make full use of the historical data as the base predictions. Lastly, a self-supervised task named stochastic augmentation is proposed to pre-train our ST-TSNet to learn spatial-temporal representations and fine-tune them to benefit the prediction task.

The contributions of this work are summarized as follows.

- We propose a novel framework Spatial-temporal Transformer Network with Self-supervised Learning (ST-TSNet) to capture spatial-temporal features.
- We employ a simple yet effective skip connection strategy, plugged into ST-TSNet, to make full use of the temporal similarities in traffic flow data.
- We introduce self-supervised learning to our framework and design a pre-training task called stochastic augmentation to explore spatial-temporal features to boost traffic flow prediction task.
- We conduct extensive experiments on two benchmarks (TaxiBJ and TaxiNYC) to evaluate the effectiveness of our methods and the results show that our ST-TSNet outperforms state-of-the-art methods.

2. Related Work

Traffic Flow Prediction. There are two types of flow data in the traffic flow prediction task: grid-like raster data and graph data and thus two distinct paradigms are derived for the two types of data [7]. In our work, we focus on raster data. Existing mainstream traffic prediction methods for raster data fall into one of the following classes: statistical methods or deep learning methods. Statistical methods include auto-regressive integrated moving average (ARIMA) [8], Kalman filtering [9] and historical average. These methods often require strong and trivial theoretical assumptions, which may violate the nonlinearity of traffic flow data, thus having poor performance in the real world. Recent advances have witnessed the impressive capacity of deep learning to extract nonlinear features from big data [10]. Many researchers are inspired to apply deep learning to handle traffic flow prediction task. Existing deep learning methods are based on convolutional neural networks (CNNs) and recurrent neural networks (RNNs) [11]. ST-ResNet [1] first employs the CNNs with residual connections to learn the spatial dependencies and construct historical data into different branches according to the temporal semantics to learn temporal features. Similar ideas are adopted by subsequent works [2, 12] in which 3D convolution is used to learn the spatial-temporal dependencies. Moreover, RNN-based models [13, 14] are inspired to use convolutional layer to capture spatial features and sequential hierarchy (e.g., LSTM and GRU) to extract temporal patterns. However, these methods are time-consuming as they make predictions step by step and may suffer gradient vanishing or explosion when capturing long-range sequences [15]. To alleviate the problems, [15, 16] discard the recurrent chain structure and employ Multiplicative Cascade Unit (CMU) with autoencoders while preserving the convolutional layers for learning spatial features. The methods used by existing works can be considered from spatial and sequential perspectives. From the spatial perspective, convolutional layers are the mainstream, including 2D and 3D convolution. From the sequential perspective, there are many choices, including RNN, GRU, LSTM and CMU. Most existing works are a combination of these methods. In summary, existing methods that based on CNNs suffer from short-range bias as the small receptive field limits their capacity to extract global dependencies.

Self-supervised Learning. Self-supervised learning is a great way to extract training signals from massive amounts of unlabelled data and to learn general representation to facilitate downstream tasks which the labelled data are limited. To generate supervision information from data, a general strategy is to define pre-training tasks for models [17, 18] to learn semantic representations, and then transfer them to downstream tasks to improve performance and robustness. Many works in computer vision have defined various tasks based on heuristic methods[19, 20]. For example, [21] learns visual representations by predicting the image rotations. In natural language processing (NLP), masked language modeling, e.g., Bert [6] have shown to be excellent for pre-training language models. These methods mask a portion of the input sequence and train models to predict the missing content with the rest. Such methods are effective for learning semantic correlations of elements within a sequence, e.g., sentence. The traffic flow data can also be viewed as a sequence temporally, while the effectiveness of self-supervised learning remains unexplored in traffic flow prediction task.

3. Methods

3.1. Problem Formulation

We partition a city into an image-like grid map according the longitude and latitude, as shown in the traffic flow map of Fig.1, where each grid denotes a region. The value of a grid denotes the traffic flow (inflow or outflow). The device deployed at a region will periodically record the number of people arriving at and departing from the location to collect the inflow and outflow. The traffic flow map of the entire city at time t is noted as $\mathbf{x_t} \in \mathbb{R}^{2\times H\times W}$, where 2 refers to the inflow and outflow, and H and W denote the number of rows and columns of the grid map, respectively. The purpose of traffic flow prediction is to predict x_n given historical traffic flow records $X_{his} = \{\mathbf{x}_t \mid t = 0, \ldots, n-1\}$. As shown in Fig.1, the historical data is summarized into two categories in the time axis: Closeness sequence $X_{close} = \{X_{n-1}, X_{n-2}, \cdots, X_{n-(d_c-1)}, X_{n-d_c}\} \in \mathbb{R}^{2\times d_c\times H\times W}$ is a concatenation of recent historical data where d_c is the length of closeness sequence. Trend sequence $X_{trend} = \{X_{n-L_{week}}, X_{n-2\cdot L_{week}}, \cdots, X_{n-d_t\cdot L_{week}}\} \in \mathbb{R}^{2\times d_t\times H\times W}$ is a concatenation of periodic historical data from the past few weeks, where d_t is the length of trend sequence, L_{week} is the number of intervals within a week.

3.2. Spatial-temporal Transformer Network

Overall, we employ a symmetric structure for handling the trend data X_{trend} and and the closeness data X_{close}: a Pre-Conv Block followed by a ViT with two shortcuts (i.e., two blue lines shown in Fig.1) from the input to the fusion layer. In the end, fusion layer adaptively merges four components (two residual components $\hat{X}_{rc}$ and $\hat{X}_{rt}$, two outputs $\hat{X}_{close}$ and $\hat{X}_{trend}$) to generate prediction $\hat{x}_n$.

Pre-Conv Block. The traffic flow in a region is highly relevant to its nearby regions. We design a Pre-Conv Block for capturing such short-range dependencies. As illustrated in Fig.1, Conv1 and Conv2 are the main convolutional layers to capture short-range dependencies. Thus, we employ a small kernel size (i.e., 3×3) which leads to the receptive field of 5. Such design ensures the Pre-Conv Block only captures the local dependencies at most in 5×5 regions. The short-range dependencies are well-captured by the Pre-Conv Block while leaving the long-range features to the vision transformer. Inserting CNNs before ViT has shown to be effective in strengthening the capacity of ViT [22]. Conv3 is the residual shortcut, employing 64 kernels with size 1×1, which adds up to the main branch as a residual component. Generally, we will use much more kernels (e.g., 64) than that in Conv4. By enlarging and then reducing the number of channels, Pre-Conv Block can learn various spatial-temporal dependencies and then refine them into a compact feature map.

Vision transformer. We apply vision transformer (ViT) [4] after the Pre-Conv Block to capture the global dependencies, as shown in the right of Fig.1. ViT is comprised of two main components: "Patchify" stem and transformer encoder. "Patchify" stem spatially splits the input feature map into non-overlap $p \times p$ patches and linearly projects patches into tokens. Each token contains the information of a patch of regions. Then the tokens are fused with learnable positional encoding to preserve the 2D positional information and fed into transformer encoder. The encoder utilizes a multi-head self-attention mechanism to model the long-range dependencies followed by a layer normalization and residual connection (Add & Norm) to the next sub-layer, where a Feed Forward Network (FFN) and another Add & Norm are employed to further process the tokens. Lastly, the tokens are averaged and then linearly transformed to generating output:$\hat{X}_{close}$ and $\hat{X}_{trend}$.

Skip Connection. Skip Connection are employed to transfer similar patterns from the historical observations to the output as the base prediction. To preserve the original similar patterns in historical data, we directly connect input X_c and X_t to the fusion layer, as shown in the blue line of Fig.1. Before connecting, we aggregate historical input data in the time dimension to match the shape. For two historical sequences $X_{close} \in \mathbb{R}^{2\times d_c \times H \times W}$ and $X_{trend} \in \mathbb{R}^{2\times d_t \times H \times W}$, we compute:

$$\hat{X}_{rc} = f(X_c) \in \mathbb{R}^{2\times 1 \times H \times W}, \tag{1}$$

$$\hat{X}_{rt} = f(X_t) \in \mathbb{R}^{2\times 1 \times H \times W}, \tag{2}$$

where $\hat{X}_{rc}$ and $\hat{X}_{rt}$ are the two residual components. $f(\cdot)$ is an aggregation function $\mathbb{R}^{2\times D \times H \times W} \rightarrow \mathbb{R}^{2\times 1 \times H \times W}$, where D denotes the length of historical data sequence. Here we use a summation function. Finally, the two residual components will be fused in the fusion layer.

Fusion Layer. The degree of influence of the four components (i.e., two outputs $\hat{X}_{close}$, $\hat{X}_{trend}$ and two residual components $\hat{X}_{rc}$, $\hat{X}_{rc}$) is different, and the influence in different regions also varies. Therefore, to dynamically calibrate their contributions, we follow [23] to use a parametric-matrix-based fusion method, where the parameter matrices are learned from historical data. Formally,

$$\begin{aligned}\hat{X}_{pred} =& w_c \cdot \hat{X}_{close} + w_t \cdot \hat{X}_{trend} + \\ & w_{rc} \cdot \hat{X}_{rc} + w_{rt} \cdot \hat{X}_{rt},\end{aligned} \tag{3}$$

where $\cdot$ denotes element-wise multiplication, w is the learnable parameter that measures the influence of each component.

3.3. Self-supervised Learning with Stochastic Augmentation

Our stochastic augmentation aims to pretrain our model to learn general spatial-temporal features to facilitate the prediction task. The pretraining strategy is conceptually simple: we select a group of continuous traffic frames, randomly sample a frame as the predicted target and use the rest to predict the target. Such scheme can be expanded to three cases: (1) if the last frame is selected as the target, then this is similar to supervised training, the historical records are used to predict future data; (2) if the earliest frame is the target, then future observations are used to predict the past frame, as shown in the green axis of Fig.1; (3) if any intermediate frame is selected as the target, then the historical data and future observations are used to predict present, as shown in the red axis of Fig.1. Different from the downstream prediction task, where input historical records and future data are paired to be the training samples, our stochastic augmentation produces several times more samples for pretraining by randomly constructing input-target pairs. For example, given a group of five frames, the supervised learning only gives one training sample as stated in case (1). While our stochastic augmentation paradigm yields five pretraining samples (every frame in the group is selected to be the target once), five times more samples than supervised training. With the large amount of pretraining samples, our models can explore useful spatial-temporal representations for the downstream prediction task. Specifically for the traffic flow prediction task, we define the group as the union of closeness data, trend data, and predicted ground truth: $X_{group} = X_{close} \cap X_{trend} \cap x_n$. Then we randomly sample one snapshot as the target α and the rest data $\Omega = X_{group} - \alpha$ as the input, constructing pre-training data (Ω, α) to pre-train our model. The algorithm is depicted in Alg.1.

Algorithm 1 The pre-training procedure with stochastic augmentation.

```
Input: MASA model: $f_\theta$, closeness data: $X_{close}$, trend
    data: $X_{trend}$, and predicted future data: $x_n$.
Output: pre-trained MASA model.
repeat
    $X_{group} \leftarrow X_{close} \cup X_{trend} \cup x_n$
    target $\alpha \leftarrow RandomSampling(X_{group})$
    Remaining snapshots $\Omega \leftarrow X_{group} - \alpha$

    pre-trained data $(\Omega, \alpha)$
    predictions $\hat{y} \leftarrow f_\theta(\Omega)$
    loss $\leftarrow MSELoss(\hat{y}, \alpha)$
    $backprop(loss)$
    update $f_\theta$
until stop criteria is met;
```

4. Experiments

4.1. Dataset and Evaluation

Dataset. Our experiments are based on two traffic flow datasets: TaxiBJ and TaxiNYC. Additional external data

Table 1
Performance comparison of different methods on TaxiBJ and TaxiNYC.

Model	**TaxiBJ**			**TaxiNYC**		
	RMSE	MAPE (%)	APE	RMSE	MAPE (%)	APE
HA	40.93	30.96	6.77E+07	164.31	27.19	7.94E+05
ST-ResNet [23]	17.56±0.91	15.74±0.94	4.81E+07±3.03E+05	35.87±0.60	22.52±3.43	6.57E+05±1.00E+05
MST3D [12]	21.34±0.55	22.02±1.40	4.81E+07±3.03E+05	48.91±1.98	23.98±1.30	6,98E+05±1.34E+04
ST-3DNet [2]	17.29±0.42	15.64±0.52	3.43E+07±1.13E+06	41.62±3.44	25.75±6.11	7.52E+05±1.78E+05
3D-CLoST [14]	17.10±0.23	16.22±0.20	3.55E+07±4.39E+05	48.17±3.16	22.18±1.05	6.48E+05±3.08E+04
STAR [24]	16.25±0.40	15.40±0.62	3.38E+07±1.36E+06	36.44±0.88	25.36±5.24	7.41E+05±1.53E+05
PredCNN [15]	17.42±0.12	15.69±0.17	3.43E+07±3.76E+05	40.91±0.51	25.65±2.16	7.49E+05±6.32E+04
STREED-Net [16]	**15.61±0.11**	14.73±0.21	3.22E+07±4.51E+05	36.22±0.72	20.29±1.48	5.93E+05±4.31E+04
ST-TSNet (ours)	16.04±0.08	**14.63±0.05**	**3.20E+07±1.05E+5**	**34.34±0.32**	**15.68±0.09**	**4.58E+05±2.52E+03**

Table 2
Ablation study of sub-modules in ST-TSNet.

Variant	TaxiBJ			TaxiNYC		
	RMSE	MAPE (%)	APE	RMSE	MAPE (%)	APE
ViT	20.16	34.68	7.60E+07	51.82	96.52	2.12E+08
ViT + SC	17.12±0.35	15.56±0.29	3.41E+07±6.29E+05	57.45±5.39	22.99±2.59	6.71E+07±7.57E+05
PC + SC	19.17±0.05	29.16±1.14	6.39E+07±2.50E+06	37.36±0.32	49.24±1.94	1.08E+08±4.25E+06
ViT + PC	16.34±0.21	14.70±0.13	3.22E+07±2.86E+05	37.29±2.88	16.83±0.24	4.91E+07±7.11E+04
ViT + PC + SC	16.14±0.16	**14.62±0.06**	**3.20E+07±1.38E+05**	34.87±0.39	16.18±0.20	4.72E+07±5.71E+04
ViT + PC + SC + SA	16.07±0.06	14.68±0.08	3.22E+07±1.72E05	34.47±0.23	15.90±0.08	4.64E+07±2.43E+04
ST-TSNet (w Ext)	**16.04±0.08**	14.63±0.05	3.21E+07±1.05E+05	**34.34±0.32**	**15.68±0.09**	**4.58E+07±2.52E+05**

including DayOfWeek, Weekday/Weekend, holidays, and meteorological data (i.e., temperature, wind speed, and weather) are processed into a one-hot vector. There are 20,016 constructed samples in TaxiBJ and 41,856 in TaxiNYC.

- TaxiBJ [23]: TaxiBJ is a citywide crowd flow dataset collected every half hour in Beijing. Based on the geographic area of Beijing, we partition the Beijing city into 32×32 regions.
- TaxiNYC [16]: TaxiNYC is the taxi trip record dataset collected every one hour in New York City. New York City is divided into 16×8 regions based on the longitude and latitude[1].

Evaluation Metric. Three metrics: Rooted Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and Absolute Percentage Error (APE) are used to evaluate our proposed method. We follow previous works that compute the metrics on traffic flow value that is larger than 10 to ensure a fair comparison. We conducted experiments ten times for reliable results and presented the means and standard variances of the results.

4.2. Implementation Details

Min-Max normalization is applied in our experiments to scale the data to range $[-1, 1]$ and denormalize the predicted target back to the original value. We split the last 28 days as the test set for both datasets, and the remaining are regarded as training data. During training, we select 90% of the training data for training models and the remaining 10% is the validation set to early-stop our training algorithm. Our model is implemented and trained by PyTorch. We use Adam [25] as the optimizer with a learning rate of 0.001 for TaxiBJ and 0.005 for TaxiNYC. Cosine learning rate decay is employed to adjust the learning rate at each iteration. The batch size is 128 for both TaxiBJ and TaxiNYC. We run our model for 600 epochs on TaxiBJ and 800 epochs on TaxiNYC. Our ViT has two blocks, and the patch size is set to $(8, 8)$; the token dimension is set to 128; the number of attention heads is 2; the size of FFN is 512.

4.3. Quantitative Comparison

Table 1 shows the comparing results against the state-of-the-art methods. We compare our ST-TSNet with the following baselines: HA, ST-ResNet [23], MST3D [12], ST-3DNet [2], 3D-CLoST [14], STAR [24], and PredCNN [15]. The results of the baselines are from [16].

On TaxiBJ, our method exceeds the SOTA STREED-Net in terms of MAPE and APE and achieves comparable results in RMSE. While on TaxiNYC, our method significantly outperforms the SOTA ST-ResNet across all metrics by a fair margin (1.53 RMSE, 4.61 MAPE, and 1.35E+05 APE improvement).

ST-TSNet has a more significant performance improvement on TaxiNYC than TaxiBJ. The possible reason of

[1]The raw records are available at the NYC government website. A processed version for experiments is available at github

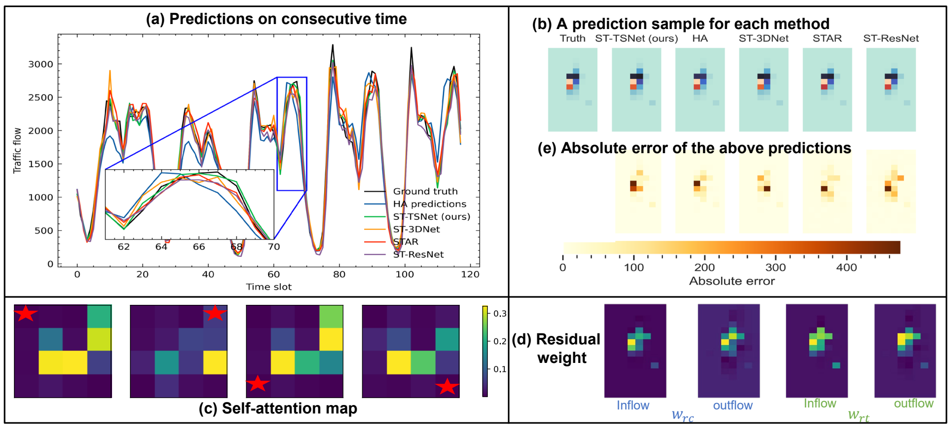

Figure 2: Qualitative analysis of our methods. (a) comparing the predicted results of each method at different time slots. (b) visualizing a prediction sample for each method and (e) showing the absolute errors of these predictions. (c) illustrating the self-attention scores of four corner patches (the pentagram-marked) for other patches and revealing that they attend to remote patches (brighter color) for long-range spatial dependencies. (d) visualizing the inflow and outflow weight of the two residual components in the fusion layer; high-flow regions usually have a higher weight.

the improvement is that the amount of data of TaxiNYC is twice that of TaxiBJ (41,856 vs. 20,016), which significantly facilitates the pre-training. This result prove the effectiveness of the self-supervised learning module proposed in our method. STREED-Net and STAR have impressive performance on TaxiBJ against other baselines due to the simple single-branch design. However, such simple architecture performs worse than ours in a larger dataset TaxiNYC (1.88 RMSE higher than our ST-TSNet) as there are rich spatial-temporal information that a single-branch structure can not extract effectively. Although STREED-Net and PredCNN both introduce cascading hierarchical structure in their backbone, STREED-Net has better performance than PredCNN. The reason is that STREED-Net additionally introduces channel and spatial attention mechanisms to dynamically refine the learned features to generate predictions. Nevertheless, the cascading hierarchical structure still suffers from short-range bias as it only allows distant snapshots to interact at higher layers. ST-ResNet, STAR, and PredCNN introduce a 2D convolutional layer, and MST3D, ST-3DNet, and 3D-CloST employ 3D convolution. The 3D convolutional layer is better than the 2D counterparts as it can additionally capture temporal features, while 2D convolutions are restricted to only capture spatial features. However, they all suffer from short-range bias due to the small receptive field of convolution. Moreover, they do not introduce the skip connection and any additional pre-training strategies, resulting in inferior performance.

4.4. Qualitative Analysis

We offer four intuitive visualizations of proposed methods to explain their behaviors in Fig.2. Fig.2 (a) compares the predictions of each method at different time intervals. The magnified subplot reveals that our method has better accuracy in predicting the peak. Fig.2 (b) spatially visualizes a prediction sample of each method, and Fig.2 (e) displays the absolute errors of these predictions, demonstrating that our ST-TSNet has lower prediction errors than baselines. Fig.2 (c) shows the self-attention map for four reference patches. The visualizations are produced by attention scores computed via query-key product in the ViT. We use the pentagram-marked regions as query, and show which patch (region) they attend to. The four corner patches usually attend to remote regions (brighter color meaning higher attention scores) while caring less about their neighbors. The reason is that the short-range features are perfectly captured and encoded into tokens by Pre-Conv Block, resulting in the ViT focusing more on the long-range features. Fig.2 (d) visualizes the weights of inflow and outflow of two residual components. Combining the ground truth in Fig.2 (c), we observe that although the weights vary in different regions and differ from inflow to outflow, they tend to concentrate on the regions with higher traffic flow. The reason is that these regions show a more regular time series, having more similar patterns in residual components.

4.5. Ablation Study

To verify the effectiveness of proposed methods, we design a list of variants by appending modules step by step and comparing them on TaxiBJ and TaxiNYC. The basic variant is Vision transformer (ViT). We separately append skip connection (SC), Pre-Conv Block (PC), and stochastic argumentation pre-training (SA) to ViT to construct other variants. We further consider the external factors on our ST-TSNet (ST-TSNet (w Ext)). We use an external module (two-layer multilayer perceptron) to model the external features according to [5]. The external data is transformed and added together with the main output to yield prediction.

The results in Table 2 show that: 1) the full version of the our methods (i.e., ST-TSNet (w Ext)) achieves the best performance. 2) Adding each module step by step will progressively improve the performance. It suggests that each module is an indispensable component for our ST-TSNet.

We additionally study the strategy of the skip connection by introducing a new residual component: the Pre-Conv Block output Y_{conv}. We investigate two connection strategies: additionally and solely connect Y_{conv} to the fusion layer. Results show that the two strategies degrade performance (1.66 and 1.35 RMSE degradation), suggesting that the Y_{conv} is harmful for prediction. The performance degradation may be caused by the convolutional operations in Pre-Conv Block disrupt the semantic information in historical data (e.g., traffic distributions), resulting in the Y_{conv} and predicted target have different distributions. In contrast, the historical records (x_{trend} and x_{close}) and the predicted target are collected from the same distribution and temporally correlated. Thus the historical records share similar patterns with the predicted target that can directly contribute to the prediction, while the Y_{conv} confuses the model.

5. Conclusion

In this paper, we present a novel traffic prediction framework, spatial-temporal Transformer Network with Self-supervised Learning (ST-TSNet) for learning spatial-temporal features. ST-TSNet is equipped with Pre-Conv Block and ViT to capture local and spatial dependencies. In addition, we observe the similarity in traffic flow data, which enables us to take advantage of the historical data as the base prediction for the future. Finally, we propose a pretext task named stochastic argumentation to enable models to further explore spatial-temporal representations under limited data. Experiments on two datasets demonstrate the superiority of our proposed methods.

Acknowledgments

This research was funded by the National Natural Science Foundation of China under Grant No.62062033, and the Natural Science Foundation of Jiangxi Province under Grant No.20212BAB202008. Zhangzhi, in particular, would like to thank his father Jianhua Peng and mother Changmei Zhang for countless love and support during the work was developed. I love you all.

References

[1] J. Zhang, Y. Zheng, D. Qi, Deep spatio-temporal residual networks for citywide crowd flows prediction, in: Proc. of AAAI, 2017.

[2] S. Guo, Y. Lin, S. Li, Z. Chen, H. Wan, Deep spatial–temporal 3d convolutional neural networks for traffic data forecasting, IEEE Transactions on Intelligent Transportation Systems (2019).

[3] S. Fang, Q. Zhang, G. Meng, S. Xiang, C. Pan, Gstnet: Global spatial-temporal network for traffic flow prediction, in: Proc. of IJCAI, 2019.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, in: Proc. of ICLR, 2021.

[5] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, Dnn-based prediction model for spatio-temporal data, Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (2016).

[6] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proc. of ACL, 2019.

[7] X. Yin, G. Wu, J. Wei, Y. Shen, H. Qi, B. Yin, Deep learning on traffic prediction: Methods, analysis and future directions, IEEE Transactions on Intelligent Transportation Systems (2021).

[8] B. M. Williams, L. A. Hoel, Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results, Journal of Transportation Engineering-asce (2003).

[9] J. hua Guo, W. Huang, B. M. Williams, Adaptive kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification, Transportation Research Part C-emerging Technologies (2014).

[10] R. Salakhutdinov, Deep learning, in: Proc. of KDD, 2014.

[11] Y. Lv, Y. Duan, W. Kang, Z. Li, F. Wang, Traffic flow prediction with big data: A deep learning approach, IEEE Transactions on Intelligent Transportation Systems (2015).

[12] C. Chen, K. Li, S. Teo, G. Chen, X. Zou, X. Yang, R. Vijay, J. Feng, Z. Zeng, Exploiting spatio-temporal correlations with multiple 3d convolutional neural networks for citywide vehicle flow prediction, 2018 IEEE International Conference on Data Mining (ICDM) (2018).

[13] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, J. Liu, Lstm network: a deep learning approach for short-term traffic forecast, Iet Intelligent Transport Systems (2017).

[14] S. Fiorini, G. Pilotti, M. Ciavotta, A. Maurino, 3d-clost: A cnn-lstm approach for mobility dynamics prediction in smart cities, 2020 IEEE International Conference on Big Data (Big Data) (2020).

[15] Z. Xu, Y. Wang, M. Long, J. Wang, Predcnn: Predictive learning with cascade convolutions, in: Proc. of IJCAI, 2018.

[16] S. Fiorini, M. Ciavotta, A. Maurino, Listening to the city, attentively: A spatio-temporal attention boosted autoencoder for the short-term flow prediction problem, ArXiv preprint (2021).

[17] R. Zhang, P. Isola, A. A. Efros, Colorful image colorization, in: Proc. of ECCV, 2016.

[18] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, A. A. Efros, Context encoders: Feature learning by inpainting, in: Proc. of CVPR, 2016.

[19] I. Misra, L. van der Maaten, Self-supervised learning of pretext-invariant representations, in: Proc. of CVPR, 2020.

[20] M. Noroozi, P. Favaro, Unsupervised learning of visual representations by solving jigsaw puzzles, in: Proc. of ECCV, 2016.

[21] S. Gidaris, P. Singh, N. Komodakis, Unsupervised representation learning by predicting image rotations, in: Proc. of ICLR, 2018.

[22] A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li, H. Shi, Escaping the big data paradigm with compact transformers, ArXiv abs/2104.05704 (2021).

[23] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, T. Li, Predicting citywide crowd flows using deep spatio-temporal residual networks, Artif. Intell. (2018).

[24] H. Wang, H. Su, Star: A concise deep learning framework for citywide human mobility prediction, 2019 20th IEEE International Conference on Mobile Data Management (MDM) (2019).

[25] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proc. of ICLR, 2015.