Temporal Fusion Transformers Model for Traffic Flow Prediction

Yuxuan Zhou

E-mail: ndzgbwdm@foxmail.com

The Hong Kong Polytechnic University, China

Abstract: Temporal Fusion Transformers (TFT) is a Transformer model for multi-step forecasting tasks. Because TFT models can integrate decoders to import various types of inputs, including static covariates, known future inputs, and other exogenous time series observed only in the past, which are well performed in the multi-step prediction of time series. To learn temporal relationships at different scales, TFT uses a cyclic layer for local processing and an interpretable self-attention layer for long-term dependence. TFT leverages specialized components to select relevant functions and inhibits unnecessary components through a series of gating layers to achieve high performance in a wide range of scenarios. When the model was proposed, it was considered to have good interpretability. As the research continues to increase, people put forward a lot of different opinions about this. This paper focuses on the explain ability of the TFT model and its attention mechanism.

Keywords: Transformer, Attention, Temporal Fusion Transformers model.

1 INTRODUCTION

Traffic flow is an important index to measure traffic conditions. Multi-step forward traffic flow prediction can help the traffic management agency to alleviate traffic congestion in advance and make the urban planning bureau more reasonable road planning [4]. It has a certain reference value for improving the prediction accuracy of multi-step leading speed [7]. However, accurate forecasting of traffic flows is particularly challenging due to the difficulty in capturing the spatio-temporal dependence of traffic data[5-6]

Previous studies have mainly focused on various traditional statistical methods [8], such as vector autoregression-based (VAR) model and comprehensive autoregression-based moving average (ARIMA) model [9]. Recently, more and more neural network-based models have been applied to the field of traffic prediction [10]. Since the prediction of traffic flow is mainly based on the temporal correlation of traffic flow series, the circulating neural network (RNN) has been widely used to solve this problem. RNN can capture the time evolution of traffic speed, especially the long-time memory (LSTM) can learn the long-time dependence of speed series. Multiple RNN's were designed to predict [11]. However, the traditional RNN model has two problems: the huge demand for computing resources in the training process and the inability to consider long-term dependence [2]. To overcome the above shortcomings of RNN, researchers have begun to study

BDEDM 2023, January 06-08, Changsha, People's Republic of China
Copyright © 2023 EAI
DOI 10.4108/eai.6-1-2023.2330350

a new structure: the attention mechanism. Attention mechanism is widely used in the field of time series prediction because it can more effectively model dependencies sequentially [12].

In order to solve the above problems, this study adopts the transformer based time fusion transformer (TFT) structure to predict the traffic speed. Unlike the above methods, TFT is able to take into account a variety of input variables, and several new architectures have been introduced in TFT to improve predictive performance. (1) TFT uses gate control module and variable shielding network to fuse time information of velocity data of different scales together. The static information encoder is used to encode the number of detectors in the data acquisition field0 [2]. (2) TFT uses sequence-sequence layer to capture short-term time correlation in traffic speed time series, and uses self-attention mechanism to capture long-term time correlation in traffic speed time series. The research results of this paper are mainly reflected in two aspects :(1) When the prediction time is 30 min, the TFT model has a good prediction accuracy for multiple lanes. (2) In practice, the explainability of TFT models is difficult to determine.

2 DATES

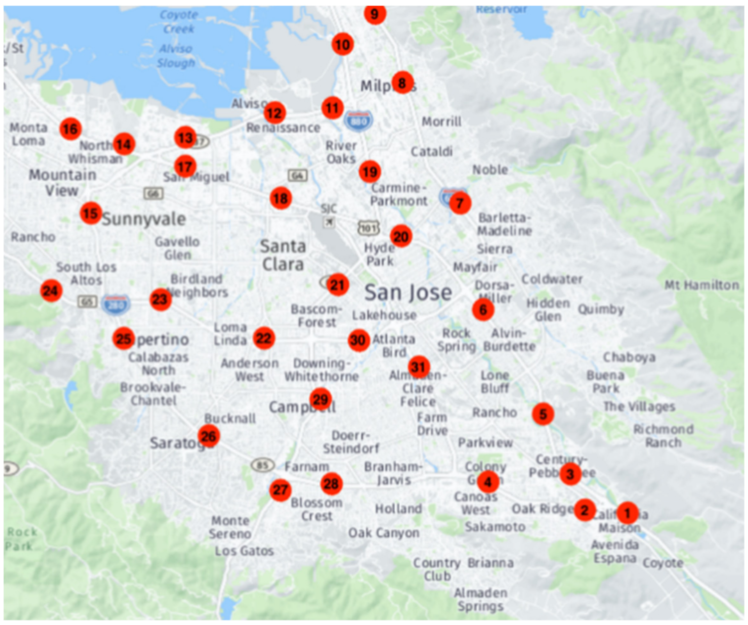


Figure 1: 31 sensor collection points

The experimental data set is comes from the PeMS system of the transportation department of California, USA. In this experiment, I selected 31 sensor collection points in the San Jose Area and collected data for a total of 91 days from 2022/1/1 to 2022/4/1 for every 5mins. The data from January 1 to March 25 were selected as the training dataset to determine the model parameters and optimize the over parameters. The data from March 26 to April 1 were selected as the test dataset to conduct performance evaluation. Missing values are averaged using adjacent values. And abnormal value use the same time value of last week.

3 METHODS

3.1Quantile output and loss functions

TFT supports quantile prediction. Let i represents unique entities in the traffic time series dataset. The definition of multi-step prediction problem can be simplified into the following formula:

$$\widehat{y_i}(q,t,\tau) = f_q\left(\boldsymbol{\tau}, \boldsymbol{y}_{i,t-k:t}, \boldsymbol{z}_{i,t-k:t}, \boldsymbol{x}_{,t-k:t}, \boldsymbol{s}_i\right) \tag{1}$$

where, $\widehat{y_i}(q,t,\tau)$ under the point in time t, the first to predict the future tau on q points a numerical. $f_q(\cdot)$ is the prediction model. $\boldsymbol{y}_{i,t-k:t}$:historical target variable. $\boldsymbol{z}_{i,t-k:t}$:Time-varying variables that can be observed Past-observed Inputs. $\boldsymbol{x}_{,t-k:t}$: Apriority-known Future Inputs. $\boldsymbol{s}_i$:Static Covariates.

TFT generates point prediction intervals by simultaneously predicting different percentiles (e.g., 10, 50, and 90) for each time step. Quantile prediction is generated using the linear transform output of TFT's decoder. Joint minimization of quantile loss was used to train TFT, and the outputs of all quantiles were added as follows:

$$\mathcal{L}(\Omega, \boldsymbol{W}) = \sum_{y_t \in \Omega} \sum_{q \in Q} \sum_{\tau=1}^{\tau_{max}} \frac{QL(y_t, \widehat{y_i}(q, t-\tau, \tau), q)}{M\tau_{max}} \tag{2}$$

$$QL(y, \hat{y}, q) = q(y-\hat{y})_+ + (1-q)(\hat{y}-y)_+ \tag{3}$$

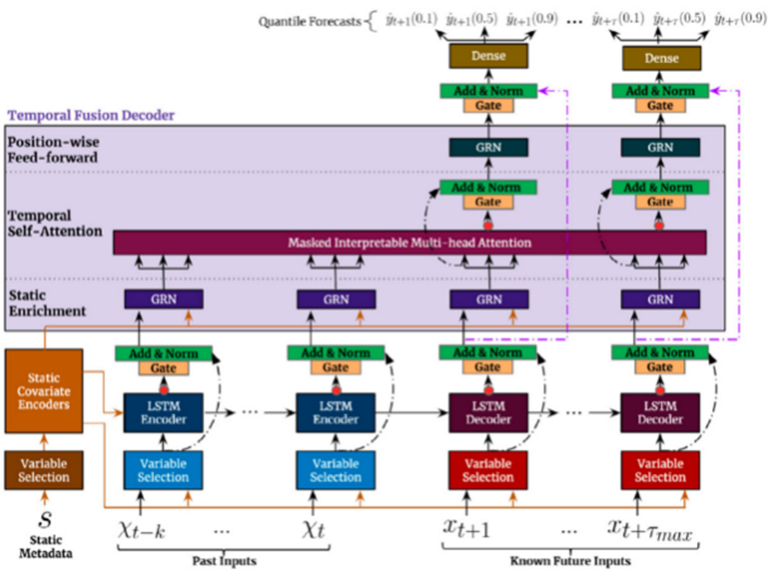
where, Ω is the training data field containing samples, $\boldsymbol{W}$ represents the weight of TFT, Q is the set of output quantiles (Q={0.1,0.5,0.9} used in the experiment). $\mathcal{L}(\Omega, \boldsymbol{W})$ is the loss of quantiles q under the average prediction point of a single sequence. In this formula, due to $(y-\hat{y})_+$ and $(\hat{y}-y)_+$ will be one negative, one positive, so the formula can be converted into:

$$QL(y, \hat{y}, q) = \max\left(q(y-\hat{y}), (1-q)(\hat{y}-y)\right) \tag{4}$$

In order to avoid the problem of inconsistent prediction dimensions under different prediction points, the author also did regularization processing, number 2 because only two quantiles of P50 and P90 are concerned here:

$$q - Risk = \frac{2\sum_{y_t \in \hat{\Omega}} \sum_{\tau=1}^{\tau_{max}} QL(y_t, \widehat{y_i}(q, t-\tau, \tau), q)}{\sum_{y_t \in \hat{\Omega}} \sum_{\tau=1}^{\tau_{max}} |y_t|} \tag{5}$$

4 MODEL ARCHITECTURE



Quantile Forecasts
$\hat{y}_{t+1}(0.1)$ $\hat{y}_{t+1}(0.5)$ $\hat{y}_{t+1}(0.9)$ ... $\hat{y}_{t+\tau}(0.1)$ $\hat{y}_{t+\tau}(0.5)$ $\hat{y}_{t+\tau}(0.9)$
Dense
Add & Norm
Gate
Temporal Fusion Decoder
Position-wise Feed-forward
GRN
Temporal Self-Attention
Masked Interpretable Multi-head Attention
Static Enrichment
Static Covariate Encoders
LSTM Encoder
LSTM Decoder
Variable Selection
s
Static Metadata
χ_{t-k} ... χ_t
Past Inputs
x_{t+1} ... $x_{t+\tau_{max}}$
Known Future Inputs

Figure 2: The framework of Temporal fusion transformers model.

4.1 GRN (Gated Residual Network):

TFT consists of four main components, namely, gate mechanism, variable selection network, static covariate encoder, time processing and multilevel prediction interval prediction. The gating mechanism, which functions to skip all unused components of the architecture, provides adaptive depth and network complexity to accommodate different data sets and scenarios. The gated residual network (GRN) can make the nonlinear calculation between variables and targets of the model more flexible. GRN contains two types of input: primary input a and optional context c.

$$GRN_{\omega}(a,c) = LayerNorm\big(a + GLU_{\omega}(\eta_1)\big) \tag{6}$$

$$\eta_1 = W_{1,\omega}\eta_2 + b_{1,\omega} \tag{7}$$

$$\eta_2 = ELU\big(W_{2,\omega}a + W_{3,\omega}c + b_{2,\omega}\big) \tag{8}$$

ELU is the Exponential Linear Unit activation function which is defined in follower equation:

$$f(x) = \begin{cases} x & if\ x > 0 \\ \alpha(exp(x) - 1) & if\ X < 0 \\ f'^{(x)} = \begin{cases} 1 & if\ x > 0 \\ f(x) + \alpha & if\ X < 0 \end{cases} \end{cases} \tag{9}$$

η_1 and η_2 are intermediate layers; $\eta_1, \eta_2 \in R^{d_{model}}$; ω is an index to denote weight sharing. Letting $\gamma \in R^{d_{model}}$ be the input the GLU is shown in following:

$$GLU_{\omega}(\gamma) = \sigma(W_{4,\omega}\gamma + b_{4,\omega}) \odot (W_{5,\omega}\gamma + b_{5,\omega}) \tag{10}$$

Where $\sigma(\cdot)$ is sigmoid activation function. $W(\cdot) \in R^{d_{model} \times d_{model}}$, $\boldsymbol{b}(\cdot) \in R^{d_{model}}$ are the weights and biases. Via the GLU, GRN can control the structure of the model and neglect the unnecessary layers. Flexibility can be provided to inhibit any architecture that is not required for a given data set. Ensure the flow of effective information. The exact relationship between exogenous inputs and targets is often unknown in advance, making it difficult to foresee which variables are relevant. In addition, it can be difficult to determine how much nonlinear processing to do, and there may be situations where a simpler model will meet our needs -- for example, when the data set is small or noisy. In order to make the model flexibly apply nonlinear processing only when needed, we propose a controlled residual network. GLU can control the degree of nonlinear contribution.

4.2 VSN (Variable Selection Network)

The variable selects the network, and the corresponding input variable is selected at each time step. The variable selection network can screen out which variables are more important to the prediction problem and also remove all noise inputs in the TFT that may affect the prediction performance.

While multiple variables may be available, their correlation and specific contributions to the output are usually unknown. TFT is designed to provide instantiated variable selection by using a variable selection network applied to static and time-dependent covariates. In addition to providing insight into which variables are most important to the prediction problem, variable selection allows the TFT to remove any unnecessary noise inputs that might negatively affect performance. Most real-world time series data sets contain features with less predictive content, so variable selection can greatly aid model performance by leveraging learning capabilities only on the most significant features [3].

Let $\xi_t^{(j)} \in R^{d_{model}}$ be the input parameter after the transformation of the j-th variable at time t when $\Xi_t = \left[\xi_t^{(1)^T}, \ldots, \xi_t^{(m_x)^T}\right]^T$ is the flattened vector of all past inputs at time t. Putting Ξ_t and an external context vector $\mathbf{c}_s$ in GRN and then through a SoftMax layer can get the Variable selection weights $v_{\chi t}$. So the equation is $v_{\chi t} = \text{Softmax}\left(GRN_{v_{\chi t}}(\Xi_t, \mathbf{c}_s)\right)$. The VSN section uses static, past and future inputs to select important features by the following equations:

$$\tilde{\xi}_t = \sum_{i=1}^{m_x} \tilde{\xi}_t \boldsymbol{v}_{\chi t}^{(i)} \tilde{\xi}_t^{(i)} \tag{11}$$

$$\tilde{\xi}_t^{(i)} = GRN\left(\xi_t^{(i)}\right) \tag{12}$$

Different VSN is used for these three inputs which the parameters are not shared.

4.3 SCE (Static Covariate Encoders)

Different types of input variables should be treated differently. In this section, TFT is designed to generate four different context vectors, $\mathbf{c}_s$, $\mathbf{c}_e$, $\mathbf{c}_c$, $\mathbf{c}_h$. In fact, SCE is using GRN function. Those four contect vectors are putting into different place in the TFD section(4). To be specific, $\mathbf{c}_s$ is used in VSN. $\mathbf{c}_c$, $\mathbf{c}_h$ are devoted to initialize LSTM. And $\mathbf{c}_e$ is used in SEL(Static Enrichment) layer in TFD section.

4.4 TFD (Temporal Fusion Decoder)

TFT refine multi-head attention in transformer-based architectures, TFT Refine multi-head attention in Transformer -based architectures, To enhance interpretability. It mainly has the following three major modules.

In time series data, important points are often identified based on the values around them, such as anomalies, points of change, or periodic patterns. TFT apply a sequence- to-sequence layer to reinforce those temporal relevance. Putting $\tilde{\xi}_{t-k:t}$ and $\tilde{\xi}_{t+1:t+\tau_{max}}$ in the LSTM encoder and decoder respectively. The input can be expressed as

$$\tilde{\phi}(t,n) = LayerNorm(\tilde{\xi}_{t+n} + GLU_{\tilde{\phi}}(\phi(t,n))) \tag{13}$$

$$\phi(t,n) \in \{\phi(t,-k), \dots, \phi(t,\tau_{max})\} \tag{14}$$

n is the position index. And $\mathbf{c}_c$, $\mathbf{c}_h$ from 4.3 are devoted to initialize the cell state and hidden state respectively for the first LSTM in the layer.

4.4.1 SEL (Static Enrichment Layer)

Since Static information usually has a significant impact on the accuracy of time series prediction, Static Enrichment Layer enhances timing feature by introducing static covariable, namely simply using GRN and input $\mathbf{c}_e$ given by the static covariable encoder.

$$\theta(t,n) = GRN_\theta(\tilde{\phi}(t,n), \boldsymbol{c}_e) \tag{15}$$

4.4.2TSL (Temporal Self-Attention Layer)

The self-Attention module can learn long-term dependencies on time series data and provide for model interpretability. In TSL, it is mainly the interpretable polycephalic self-concern layer, plus GLU. The interpretable multi-head attention used in every moment ($N = \tau_{max} + k + 1$) where $\Theta(\mathrm{t}) = [\theta(t,-k), \dots, \theta(t,\tau)]^T$

$$\boldsymbol{B}(t) = InterpretableMultiHead(\Theta(t), \Theta(t), \Theta(t)) \tag{16}$$

$$\delta(t,n) = LayNorm\big(\theta(t,n)\big) + GLU_{\delta}\big(\beta(t,n)\big) \tag{17}$$

To be more specifical, the Interpretable multi-head attention has three major parameter query, key and value and then use scale dot-product as following:

$$Attention(Q,K,V) = A(Q,K)V \tag{18}$$

$$Q \in R^{N\times d_{attn}}, K \in R^{N\times d_{attn}}, V \in R^{N\times d_V} \tag{19}$$

Where $A(\cdot)$ is a normalization function, and N is the time steps.

$$(Q,K)V = Softmax(QK^T\ \sqrt{d_{attn}}) \tag{20}$$

For the multi-head aspect, this mechanism employ different heads for different representation subspaces:

$$MultiHead(Q,K,V) = \left[H_1, \ldots H_{m_H}\right]W_H \tag{21}$$

$$H_h = Attention(QW_Q^{(h)}, KW_K^{(h)}, VW_V^{(h)}) \tag{22}$$

Where $W_Q^{(h)} \in R^{d_{model}\times d_{attn}}$, $W_Q^{(h)} \in R^{d_{model}\times d_{attn}}$, $W_V^{(h)} \in R^{d_{model}\times d_V}, W_H \in R^{(m_H\cdot d_V)\times d_{model}}$, W_H is the Linear combinations of all the heads H_h.

Attention weights separately can't indicate the importance of a particular feature, so TFT modify multi-head attention to share values in each head and employ additive aggregation of all heads. In other words, For V is the multi-head shared parameter, for Q and K are the multi-head independent parameters, then calculate the multi-head attention score weighted V, and sum the average output.

$$InterpretableMultiHead(Q,K,V) = \widetilde{H}\ W_H \tag{23}$$

$$\begin{aligned} \widetilde{H} = \tilde{A}(Q,K)VW_V &= \left\{\frac{1}{m_H}\sum_{h=1}^{m_H} A\left(QW_Q^{(h)}, KW_K^{(h)}\right)\right\} V\ W_V \\ &= \frac{1}{m_H}\sum_{h=1}^{m_H} Atention\,(QW_Q^{(h)}, KW_K^{(h)}, VW_V) \end{aligned} \tag{24}$$

4.4.3PFL（Position-wise Feed-forward Layer）

Apply additional nonlinear processing to the output of the self-focused layer.

$$\psi(t,n) = GRU_{\psi}\big(\delta(t,n)\big)$$
$$InterpretableMultiHead(Q,K,V) = \tilde{H}\,W_H \tag{25}$$

$$\tilde{\psi}(t,n) = LayerNorm(\tilde{\phi}(t,n) + GLU_{\tilde{\psi}}(\psi(t,n)) \tag{26}$$

5 RESULTS

Table 1: Hyperparameter search ranges.

Hyperparameter	Range
State size	10, 20, 40, **80**, 160, 240, 320
Dropout rate	0.1, 0.2, **0.3**, 0.4, 0.5, 0.7, 0.9
Minibatch size	20, 30, 40, **50**, 64, 128
Learning rate	0.001, **0.002**, 0.004, 0.008, 0.01
Heads	1, 2, 3, **4**

Bold values indicate the optimal hyperparameters.

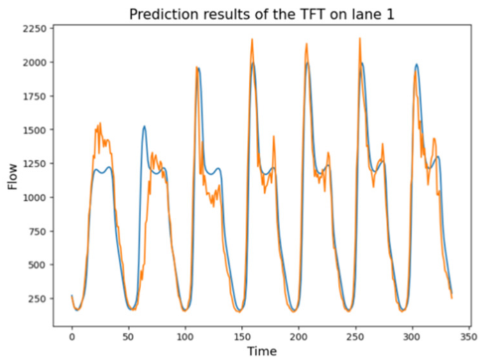


Figure 3: Prediction results of the TFT on lane 1

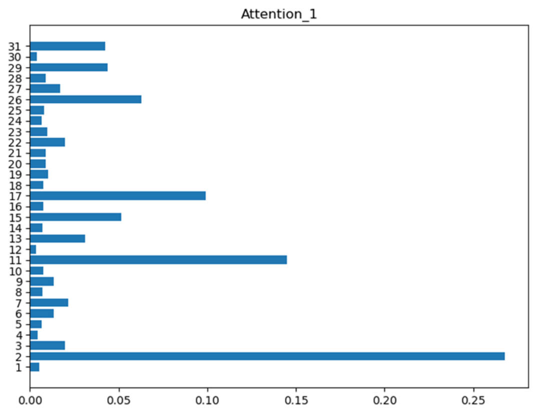


Figure 4: Attention score of TFT on lane 1

Table 2: Prediction results of different models.

	model 1 base	model 1-max	model 1-min
MAE	122.4416	1112.1327	1092.2562
RMSE	190.3704	1189.4219	1170.6600
MAPE	0.1791	1.9337	1.9264
R2 score:	0.8751	-3.8746	-3.7220
	model 3 base	model 3-max	model 3-min
MAE	175.5456	498.4471	507.2493
RMSE	276.9062	643.5049	663.3742
MAPE	0.1816	0.3482	0.3395
R2 score:	0.8723	0.3103	0.2670
	model 14 base	model 14-max	model 14-min
MAE	148.3931	170.6433	131.4054
RMSE	199.7648	218.4773	189.1805
MAPE	0.1483	0.1768	0.1381
R2 score:	0.9491	0.9392	0.9544
	model 17 base	model 17-max	model 17-min
MAE	132.7735	156.7910	138.8228
RMSE	180.2207	197.7319	174.6579
MAPE	0.1379	0.1738	0.1509
R2 score:	0.9537	0.9442	0.9565
	model 19 base	model 19-max	model 19-min
MAE	126.5685	151.6295	143.8309
RMSE	277.9803	305.3027	275.6557
MAPE	0.0956	0.1200	0.1093
R2 score:	0.9252	0.9098	0.9265
	model 26 base	model 26-max	model 26-min
MAE	122.4416	100.8592	97.6148
RMSE	190.3704	157.5504	159.8302
MAPE	0.1791	0.1273	0.1283
R2 score:	0.8751	0.9145	0.9120

6 CONCLUSIONS

In order to reduce the uncertainty of prediction results, for each lane, TFT was run 10 times and the model with the smallest loss was selected. Then, the first and last lanes with the highest importance of encoder variables were excluded respectively for comparison experiment. Each run of the model predicts a different outcome. That is because deep learning is a random machine learning algorithm. On the one hand, when training the neural network, the weight of the neural network is initialized and random.

On the other hand, this study sets the exit rate in the algorithm to avoid overfitting. Setting the exit rate means that the algorithm will randomly discard some neurons and network nodes during training. It can be seen from the test results that in multiple modelling (more than 300 times), the performance of the model is completely different, and its loss and attention score are also completely different. Further analysis shows that after removing the lane data with the most and least weight, the loss change of the model cannot be analysed and it is difficult to find out the rule. The predictive power of its models also fails to find patterns. Here, I think the main reason is the randomness of deep learning. In multiple learning, the parameters of the model obtained each time are completely different, which is difficult to reproduce. Therefore, it is difficult to reproduce the attention score of each model. To sum up, the explanatory power of its model cannot be demonstrated stably for more than 300 hours. In theory, the attention mechanism should be able to express certain explanatory ability, but in practice, it is impossible to know whether the model calculated next time will be better and more suitable to the data. The calculation difficulty is too complicated, and the calculation power needs are too large.

In many experiments, the TFT model has high predictive performance and can better simulate the time correlation through the self-attention mechanism, which provides experience for the establishment of a good long-term forecast model. And the explanatory prediction process can be given. However, it is difficult to determine whether this model is an optimal model or an interpretable model. On the other hand, TFT requires a lot of data to train the model to achieve good predictive performance and requires a lot of time and computing power. Since graph neural network can capture spatial correlation, TFT and graph neural network can be combined to predict the traffic state of road network. Another interesting area of research is the integration of traffic data, geographic information, weather information and other information into speed forecasting.

REFERENCES

[1] YAN Xu, FAN Xiao-liang, ZHENG Chuan-pan, et al. Urban traffic flow prediction algorithm based on graph convolutional neural networks [J]. Journal of Zhejiang University (Engineering Science), 2020, 54(6):9.

[2] Hao Zhang, Yajie Zou, Xiaoxue Yang, Hang Yang, A temporal fusion transformer for short-term freeway traffic speed multistep prediction, Neurocomputing, Volume 500,2022, Pages 329-340,ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2022.05.083.

[3] Lim B , Arik S O , Loeff N , et al. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting:, 10.48550/arXiv.1912.09363[P]. 2019.

[4] E.I. Vlahogianni, J.C. Golias, M.G. Karlaftis, Short-term traffic forecasting: Overview of objectives and methods, Transp. Rev. 24 (2004) 533–557, https:// doi.org/10.1080/0144164042000195072.

[5] X. Yin, G. Wu, J. Wei, Y. Shen, H. Qi, B. Yin, Deep learning on traffic prediction: methods, analysis and future directions, IEEE Trans. Intell. Transp. Syst. (2021) 1–15.

[6] Y. Wu, H. Tan, L. Qin, B. Ran, Z. Jiang, A hybrid deep learning based traffic flow prediction method and its understanding, Transp. Res. Part C Emerg. Technol. 90 (2018) 166–180.

[7] E.I. Vlahogianni, M.G. Karlaftis, J.C. Golias, Short-term traffic forecasting: Where we are and where we're going, Transp. Res. Part C Emerg. Technol. 43 (2014) 3–19.

[8] M.C. Tan, S.C. Wong, J.M. Xu, Z.R. Guan, P. Zhang, An aggregation approach to short-term traffic flow prediction, IEEE Trans. Intell. Transp. Syst. 10 (2009) 60–69

[9] Y. Zou, X. Hua, Y. Zhang, Y. Wang, Hybrid short-term freeway speed prediction methods based on periodic analysis, Can. J. Civ. Eng. 42 (2015) 570–582

[10] J. Artin, A. Valizadeh, M. Ahmadi, S.A.P. Kumar, A. Sharifi, Presentation of a novel method for prediction of traffic with climate condition based on ensemble learning of neural architecture search (NAS) and linear regression, Complexity 2021 (2021)

[11] Z. Cui, R. Ke, Z. Pu, Y. Wang, Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction, (2018)

[12] S. Al-Janabi, A.F. Alkaim, Z. Adel, An Innovative synthesis of deep learning techniques (DCapsNet & DCOM) for generation electrical renewable energy from wind energy, Soft Comput. 24 (2020) 10943–10962

[13] Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, Tomas Pfister. "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting", International Journal of Forecasting, 2021