The Sequence-to-Sequence Architecture with An Embedded Module for Long-Term Traffic Speed Forecasting with Missing Data

Ge Zheng\(^1\), Wei Koong Chai\(^1\), Vasilis Katos\(^1\)

\(^1\)Department of Computing and Informatics, Bournemouth University, Poole, Dorset, BH12 5BB, U.K.

**Abstract**—Traffic forecasting plays a crucial role in Intelligent Transportation Systems (ITSs), which is proposed to provide traffic status in advance for road users to avoid traffic congestion or other traffic incidents and for authorities to optimise the strategies of traffic management. In this paper, we develop a novel deep learning framework, based on the Sequence-to-Sequence architecture with an embedded module, for long-term traffic speed forecasting with missing data and providing high forecasting accuracy. The embedded module uses Graph Convolution Neural Network for the local spatial dependency analysis by conducting convolutional operation on the $k$-hop neighbourhood matrix, while utilises Transformer for the global spatial dependency analysis by assigning individual weights to neighbour detectors for contributing to the targeted detector. The sequence-to-sequence architecture is built to analyse temporal dependencies of the spatially-fused time series from the embedded module. To evaluate the proposed model against existing well-known ones, the real traffic speed dataset with missing data and frequent traffic incidents is used to train and test the models. The experimental results indicate that our proposed framework achieves the most accuracy forecasting, even obtaining more than 80\% accuracy for forecasting two hours in advance.

**Index Terms**—long-term, intelligent transportation system, deep learning, large-scale road networks

I. INTRODUCTION

With the sharp growth of population and vehicles, transportation infrastructures are facing huge challenges, such as serious traffic congestion, increasing number of incidents and severe delay in actual travel time. These challenges bring many problems to cities like air pollution and waste of energy. To overcome the challenges and resolve these problems, the concept of Intelligent Transportation Systems (ITSs) [1] has been proposed. It aims to offer innovative services relating to different modes of transport and traffic management based on the most advanced technologies and enable road users to be better informed and safer during travelling [2]. As an important element of ITSs, traffic forecasting is to forecast traffic status for the future based on historical traffic data and then provide it for road users to improve traffic efficiency.

Many works in traffic forecasting have been reported in the literature, including studies on traffic flow, speed and travel time, since the earliest work, that used the well-known Auto-Regressive Moving Average (ARMA) model to forecast traffic volume and occupancy, was published in late 1970s [3]. For example, some variants of ARMA like Auto-Regressive Integrated Moving Average (ARIMA) [4] and Seasonal ARIMA (SARIMA) [5] were proposed for improving the capability of traffic forecasting. ARMA and its variants are statistic models and usually used for time series forecasting with linear relationship. However, [6] pointed that a non-linear relationship exists in traffic data, and ARMA and its variants are unable to completely analyse the non-linear relationship of traffic data. Owning to the wide popularity of machine learning technologies, machine learning models with non-linear kernels or activation functions have been used for analysing non-linear relationships of traffic data and achieved better performance compared to ARMA and its variants. For example, Support Vector Regression with RBF kernel (SVR) was used for traffic flow forecasting [7] while an enhanced K-Nearest Neighbor (K-NN) algorithm was also used for traffic flow forecasting [8].

With the availability of new sensor technologies and advanced big data analysis technologies, mass data with more features is available for traffic forecasting. Therefore, deep learning models that are capable of analysing big data with high dimensions have been used for traffic forecasting, instead of machine learning models that are too shallow for dealing with high dimensional data. In 2014, Huang et al. [9] designed a deep learning model for traffic forecasting which uses a network with Deep Belief Network (DBN) model at the bottom for unsupervised pre-training and a Multi-Task Layer (MTL) at the top for unsupervised forecasting. In the same year, Lv et al. [10] built a Stacked AutoEncoder model (SAE) to learn traffic flow features and trained it in a greedy layerwise fashion. Zhang [11] developed a Convolutional Neural Network (CNN) for learning traffic data as images and then used a fully-connected layer for final forecasting. The advantage of this model is that spatial dependencies can be extracted through the kernels of CNN with non-linear activation functions and its kernel size decides the area coverage of the traffic data image. However, these models are unable to analyse the temporal dependencies of traffic data. Therefore, the Recurrent Neural Network (RNN) and its variants (Long-Short Term Memory, LSTM, and Gated Recurrent Unit, GRU) [12] , that take the advantage of analysing long-term dependencies, have been combined with CNN for traffic forecasting. Wu et al. [13] used a 1D CNN and two LSTMs to build a hybrid deep learning method...
The traffic forecasting problem on large-scale road networks, aiming to provide more accurate forecasting on large-scale road networks. The problem can be formulated as Eq. (2).

\[ \hat{X}' = F(X; \mathcal{G}(\mathcal{V}, \mathcal{E}, \tilde{A}^k)) \]  

where the objective is to learn the mapping function \( F(\cdot) \) and compute the traffic data in the next \( T' \) time steps based on the historical traffic data in \( T \) previous time steps and the road graph \( G \). For fitting the training phase, both input \( X \) and targeted \( X' \) need to be formatted to \( X \in \mathbb{R}^{B \times T \times N} \) and \( X' \in \mathbb{R}^{B \times T' \times N} \) where \( B \) is the batch size.

### III. A Novel Proposed Framework

Fig. 1 presents the framework of the proposed deep learning model, named GCNT-Seq2Seq, that is designed to comprehensively analyse the spatial and temporal dependencies of traffic data. GCNT-Seq2Seq is developed under the sequence-to-sequence architecture consisting of an encoder and a decoder for the long-term dependency analysis. The encoder is used to learn the historical information and encodes it to a context vector. The decoder is utilised to decode the context vector to the final forecasting. The modules of the sequence-to-sequence architecture are built based on graph convolutional neural networks and the transformer for analysing spatial dependencies from original traffic data and the road graph data. The following content will explain the GCNT-Seq2Seq model on the spatial dependency analysis and the temporal dependency analysis in detail, respectively.

**GCNT** as the embedded module of the sequence-to-sequence architecture is used to analyse the spatial dependencies. Fig. 2 displays the GCNT module that consists of \( L_g \) graph convolutional neural (GCN) layers and \( L_s \) transformer layers in parallel. Considering the \( l_g\text{-th} \) GCN layer at the

---

**Note:** The text contains mathematical expressions and notation related to graph theory and traffic forecasting. The complete context and details of the models and equations are beyond the scope of this summary.
of embedded features for

d\text{GCN}\text{ layers can be considered as local spatial\ features using Eq. (3). Due to \(V\) values\ GCN layer and is treated as the input of the\ l\text{hop}\ \text{layer and a linear layer. Considering the\ layer consists of three fully-connected layers, an attention\ with neurons contributing to the targeted detector, spatial features obtained here are considered as global spatial features.} 

\[ GCN_{l_g,t} = ReLU \left( (W_{g,t} \ast \tilde{A}^k)GCN_{(l_g-1),t} \right) + GCN_{(l_g-1),t} \] 

where \(GCN_{(l_g-1),t} \in \mathbb{R}^{B \times N}\) is the output of the \((l_g-1)\text{th}\) GCN layer and is treated as the input of the \(l_g\text{th}\) GCN layer, \(GCN_{l_g,t} \in \mathbb{R}^{B \times N}\) is the output of the \(l_g\text{th}\) GCN layer and the input of the \(1\text{st}\) GCN layer is \(x_t \in \mathbb{R}^{B \times N}\). \(W_{l_g,t} \in \mathbb{R}^{N \times N}\) is the weight matrix of \(\tilde{A}^k\) and \(ReLU\) is the activation function of GCN layers.

Meanwhile, transformer layers analyse spatial dependencies from the other aspect by the attention mechanism to assign individual weights to neighbours of the targeted detector so as to contribute to the targeted detector. Each transformer layer consists of three fully-connected layers, an attention layer and a linear layer. Considering the \(l_g\text{th}\) transformer layer at the \(t\text{th}\) time step as an example, three fully-connected layers are used to generate the multi-head inputs of queries \(Q_{l,t} \in \mathbb{R}^{B \times (Hd_g) \times N}\), keys \(K_{l,t} \in \mathbb{R}^{B \times (Hd_k) \times N}\) and values \(V_{l,t} \in \mathbb{R}^{B \times (Hd_v) \times N}\) by Eq. (4), in which \(H\) is the number of multi-heads and \(d_g, d_k\) and \(d_v\) are the number of embedded features for \(Q_{l,t}\), \(K_{l,t}\) and \(V_{l,t}\), respectively. The spatial weight matrix is generated by multiplying the transposition of \(K_{l,t}\) by \(Q_{l,t}\) and then utilised to obtain spatial features using Eq. (5). Due to the spatial weight matrix

\[ S_{l,t} = ReLU \left( W_{l,t} \left( \text{softmax} \left( \frac{Q_{l,t}K_{l,t}^\top}{\sqrt{d_k}} \right) \right) V_{l,t} + b_{l,t} \right) + S_{(l_g-1),t} \] 

\[ \frac{Q_{l,t}K_{l,t}^\top}{\sqrt{d_k}} \in \mathbb{R}^{B \times H \times N \times N} \] 

\[ Q_{l,t} = W_{q_l,t}^sS_{(l_g-1),t}^t \] 

\[ K_{l,t} = W_{k_l,t}^sS_{(l_g-1),t}^t \] 

\[ V_{l,t} = W_{v_l,t}^sS_{(l_g-1),t}^t \] 

\[ S_{l,t} = ReLU \left( W_{l,t} \left( \text{softmax} \left( \frac{Q_{l,t}K_{l,t}^\top}{\sqrt{d_k}} \right) \right) V_{l,t} + b_{l,t} \right) + S_{(l_g-1),t} \]

where \(S_{(l_g-1),t}^t \in \mathbb{R}^{B \times N}\) is the transposition of the output of the \((l_g-1)\text{th}\) transformer layer as the input of the \(l_g\text{th}\) transformer layer. \(S_{l,t}\) is the output of the \(l_g\text{th}\) transformer layer and the input of the \(1\text{st}\) transformer layer is \(x_t\). \(W_{q_l,t}^s\), \(W_{k_l,t}^s\) and \(W_{v_l,t}^s\) are weight matrices of queries, keys, and values, respectively. \(W_{l,t}^s\) is the weight matrix of the linear layer and \(b_{l,t}\) is the related bias. \(ReLU\) is an activation function.

Finally, the output of the last GCN layer, \(GCN_{L_g,t}\), and the output of the last transformer layer, \(S_{L_g,t}\), are concatenated, and then pass a linear layer to generate local-global spatial features \(GCNT_{L_g,t}\) using Eq. (6). In addition, the residual connection network [20] is used in each GCN and transformer layer to ensure the stable training and also supplement the important information hidden in negative values that are neglected by the \(ReLU\) activation function.

\[ GCNT_{L_g,t} = W_{L_g,t} \text{concat}(GCN_{L_g,t}, S_{L_g,t}) + b_{L_g,t} \] 

where \(W_{L_g,t}\) and \(b_{L_g,t}\) are the weight matrix and the bias, respectively.

\textbf{Seq2Seq} consists of LSTMs as the encoder and the decoder, which is used to embed our GCNT module for extracting and delivering temporal features from spatially-fused features. Taking the \(t\text{th}\) time step as an example, the encoder takes
the GCNT_{LR,t} as the input and encodes the spatially-fused features using Eq. (7).

\[
\begin{align*}
    f_t &= \sigma_g(w_f \cdot GCNT_{LR,t} + u_f \cdot h_{t-1} + b_f) \\
    i_t &= \sigma_g(w_i \cdot GCNT_{LR,t} + u_i \cdot h_{t-1} + b_i) \\
    o_t &= \sigma_g(w_o \cdot GCNT_{LR,t} + u_o \cdot h_{t-1} + b_o) \\
    c_t &= f_t \odot c_{t-1} + i_t \odot \sigma_c(w_c \cdot GCNT_{LR,t} + u_c \cdot h_{t-1} + b_c) \\
    h_t &= o_t \odot \sigma_h(c_t)
\end{align*}
\]

where \(w_f, w_i, w_o\) and \(w_c\) are the weights of the forget gate \(f_t\), the input gate \(i_t\), the output gate \(o_t\) and the cell state \(c_t\) respectively while \(b_f, b_i, b_o\) and \(b_c\) are the corresponding biases for each gate and the cell state. Furthermore, \(u_f, u_i, u_o\) and \(u_c\) are the weights of the last hidden state \(h_{t-1}\). \(\sigma_g\) denotes a sigmoid function \((\frac{1}{1+e^{-x}})\) in three gates and the operator \(\odot\) denotes Hadamard product. \(\sigma_c\) and \(\sigma_h\) are hyperbolic tangent function \((\tanh(x))\) for the cell state and the final output. \(h_t\) is considered as the output of the encoder and carries historical information. In the decoder, the output of the encoder \(h_t\) is treated as the initialised hidden state and \(GCNT_{LR,t}\) is still considered as the input. The output of the decoder is the final forecasting.

IV. EXPERIMENTS

A. Data Description

Dataset used for evaluating the proposed model and comparing our model against well-known existing models is collected from the real-world road network, named METR-LA [21]. Fig. 3 displays the locations of loop detectors in METR-LA. It includes 207 detectors and covers 4 months of traffic speed data from 1st of March to 30th of June in 2012. A time step is 5 minutes and the number of observed traffic data points is 7,094,304 (= 34272 × 207). This dataset misses some data points so that it brings more challenges. To evaluate and validate the robustness of the proposed model, discontinuous missing data points are replaced by the mean of the last and next traffic data and continuous missing data points are set as zeros to simulate traffic incidents such as traffic accidents and congestion. In addition, an undirected road graph with edge weights is used to construct the neighbourhood matrix. The pairwise road distances between detectors are first computed and then a thresholded Gaussian Kernel in [22] is utilised to build the weighted matrix.

B. Performance Metrics

To measure the results, three types of metrics named Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Root-Mean Square Error (RMSE) in the literature [14][17] are used and computed as Eq. (8), Eq. (9) and Eq. (10).

\[
MAE = \frac{1}{N \times T'} \sum_{i=1}^{N} \sum_{t=1}^{T'} |x_i^t - \tilde{x}_i^t|
\]

\[
MAPE = \frac{1}{N \times T'} \sum_{i=1}^{N} \sum_{t=1}^{T'} \frac{|x_i^t - \tilde{x}_i^t|}{x_i^t} \times 100%
\]

\[
RMSE = \left[ \frac{1}{N \times T'} \sum_{i=1}^{N} \sum_{t=1}^{T'} (x_i^t - \tilde{x}_i^t)^2 \right]^{\frac{1}{2}}
\]

where MAE presents the average absolute difference between the forecasted and real traffic speed. It is used to measure absolute forecasting error. MAPE is the relative difference between the forecasted and real traffic speed and is utilised to measure relative forecasting error. RMSE is the standard deviation of the residuals where residual is the difference between forecasted and real traffic speed.

C. Parameter Settings

To optimise the GCNT-Seq2Seq model and obtain high forecasting accuracy, there are several hybrid parameters that need to be tuned including historical time steps \(T\), targeted time steps \(T'\), the number of multi-heads \(H\), the number of embedded features \(\{d_q, d_k, d_v\}\), the learning rate \(r\), batch size \(B\) and the number of epochs. Based on the well-known existing works, the historical time steps \(T\), targeted time steps \(T'\), the number of multi-heads \(H\), the number of embedded features \(\{d_q, d_k, d_v\}\), and batch size \(B\) are set as 12, 24, 8, \{8, 8, 8\} and 32, respectively. For the learning rate setting, generally, a too small learning rate will make a training algorithm converge slowly while a too large learning rate will make the algorithm diverge. Therefore, finding an optimal learning rate is very important to improve the performance of the algorithm. However, the experimental method to find the best learning rate is time-consuming. In our work, the method named Cyclical Learning Rates (CLR) in [23] is used to optimise the learning rate as \(1.02e^{-0.3}\). In addition, stop early strategy is used to optimise the number of epochs and avoid the problem of over-fitting. It means the training process will be stopped when the training loss continues to decrease in 10 epochs while the validation loss increases. Finally, we follow the convention and use 70% of the dataset for training, 10% for validation and 20% for testing.
Fig. 4: (Color Online) Real (black color) and forecasted (other colors) traffic speed (miles/h) during a day with 288 ($\frac{1\text{day} \times 24\text{hours} \times 60\text{minutes}}{5\text{minutes}}$) time intervals from all models on METR-LA with a time interval = 5 minutes. The x-axis represents the time and the y-axis is traffic speed (miles/hour). The forecasting horizons are 30 minutes in (a) and 120 minutes in (b), respectively. The red lines in (a) and (b) are forecasted traffic speed from our GCNT-Seq2Seq model.

![Fig. 4: Real (black) and forecasted (other colors) traffic speed data](image)

**TABLE I**: Experimental results from all models on METR-LA

<table>
<thead>
<tr>
<th>Model Name</th>
<th>MAE($T'=$6/12/18/24)</th>
<th>MAPE($T'=$6/12/18/24)</th>
<th>RMSE($T'=$6/12/18/24)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRU</td>
<td>5.0715 / 6.2108 / 7.0753 / 7.7387</td>
<td>12.67% / 15.64% / 18.17% / 20.03%</td>
<td>9.1238 / 10.6058 / 11.7414 / 12.4671</td>
</tr>
<tr>
<td>CNN-LSTM</td>
<td>5.2192 / 6.4425 / 7.5352 / 8.3240</td>
<td>12.70% / 16.03% / 18.79% / 20.96%</td>
<td>9.3857 / 10.9365 / 12.1356 / 12.8291</td>
</tr>
<tr>
<td>GCNT-Seq2Seq</td>
<td><strong>4.9956 / 6.1774 / 6.9358 / 7.5813</strong></td>
<td><strong>12.10% / 15.38% / 17.44% / 19.41%</strong></td>
<td><strong>9.2731 / 10.8090 / 11.6876 / 12.3108</strong></td>
</tr>
</tbody>
</table>

![Fig. 5: The forecasting accuracy (100%-MAPE) from all comparison models](image)

**D. Results and Discussion**

To evaluate the proposed model, six baselines are used for comparison experiments, including 1) one linear feature-based model, ANN [24]; 2) two temporal feature-based models, LSTM [25] and GRU [26]; 3) one spatial feature-based model, CNN-FC [27]; 4) two spatial and temporal feature-based models, CNN-LSTM [14] and TGC-LSTM [17].

Fig. 4 presents the real (black color) and forecasted (other colors) traffic speed data in a day from the GCNT-Seq2Seq model and other six baselines. Fig. 4 (a) and (b) consider 30 minutes and 120 minutes as forecasting horizons, respectively. Overall, GCNT-Seq2Seq can efficiently forecast the trend of traffic speed changes, even for longer forecasting horizons. Besides, GCNT-Seq2Seq also capture the sudden changes caused by mission data or traffic incidents.

**TABLE I** presents experimental results from all competition models for different forecasting horizons. $T' = \{6, 12, 18, 24\}$ are responding to consider $\{30, 60, 90, 120\}$ minutes as forecasting horizons. It is observed clearly that the proposed model, GCNT-Seq2Seq, achieves the best performance among all competition models. Its MAEs, MAPEs and RMSEs for all forecasting horizons are almost the lowest. It means the forecasting accuracy of GCNT-Seq2Seq is the highest, which can be also observed in Fig. 5 that displays the forecasting accuracy (100%-MAPE) of all models. From Fig. 5, the differences of forecasting accuracy between GCNT-Seq2Seq and other models become more obvious over longer forecasting horizons. It means the superiority of our model is more obvious for longer forecasting horizons. Even for 120 minutes as forecasting horizon, the forecasting accuracy of GCNT-Seq2Seq is still more than 80% and it is also the only one model achieving over 80% accuracy among all comparison models.
models. From TABLE I, the linear feature-based model, ANN, that relies on the linear relationship of traffic speed data in time domain for forecasting almost obtains the worst results excluding 30 minutes as the forecasting horizon where the spatial feature-based model, CNN-FC, is the worst one. The probable reason is that linear relationships for longer forecasting horizons are not obvious while the spatial dependencies among road network become more important. Therefore, CNN-FC model that mainly focuses on spatial feature extraction by the convolution kernels performs better than ANN for longer forecasting horizons. Between two temporal feature-based models, LSTM and GRU, GRU always outperforms LSTM for all different forecasting horizons. This conclusion was confirmed in [12] by numerous experiments. Both CNN-LSTM and TGC-LSTM achieve similar results, because they are able to analyse both spatial and temporal features for final forecasting.

V. Conclusion

In this paper, the problem of forecasting long-term traffic on large-scale road network is addressed and a novel deep learning framework, named GCNT-Seq2Seq, is developed based on the most advanced technologies. The proposed framework takes the advantages of Graph Convolution Network (GCN) and Transformer on the different spatial dependency analysis and the advantage of the Sequence-to-Sequence (Seq2Seq) architecture on the temporal dependency analysis. GCN is used to extract the local spatial features by operating convolution on the k-hop neighbourhood matrix and Transformer is utilised to capture the global spatial features by assigning individual weights to neighbours of the targeted detector so as to contribute to the targeted traffic. The concatenation of local and global spatial features is embedded into the Seq2Seq architecture for temporal feature extraction and final forecasting. The proposed framework is compared to the existing well-known models using the real world dataset with missing data and frequent traffic incidents. Among all the models being compared, our proposed model shows the best performance and can even achieve more than 80% accuracy for forecasting two hours traffic status in advance.

REFERENCES