**Transportation Research Record**

**Short-Term Traffic Prediction Using A  Spatial-Temporal  CNN Model With Transfer Learning**

--Manuscript Draft--

<table>
<thead>
<tr>
<th>Full Title:</th>
<th>Short-Term Traffic Prediction Using A Spatial-Temporal CNN Model With Transfer Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract:</td>
<td>Advanced neural network models have recently achieved competitive results for short-term traffic prediction. However, two problems require further study: 1. The effects of the different structured convolutional neural networks (CNN); and 2. The possibility of transfer learning techniques on traffic prediction problems. The unsuitable model structure could lead to inappropriate feature learning, and insufficient data could cause training difficulties. This paper investigates the influence of different CNN structures on a traffic model and how fine-tuning of the model affects the outcomes. The results show that a well-designed CNN structure can learn better traffic features, and fine-tuning the model benefits the situation which has little training data. This paper provides a positive reference for the structural design of CNN and verifies the effectiveness of the fine-tuning method on short-term traffic forecasting tasks.</td>
</tr>
<tr>
<td>Manuscript Classifications:</td>
<td>Operations and Traffic Management; Artificial Intelligence and Advanced Computing Applications ABJ70; Artificial Intelligence</td>
</tr>
<tr>
<td>Manuscript Number:</td>
<td></td>
</tr>
<tr>
<td>Article Type:</td>
<td>Presentation</td>
</tr>
<tr>
<td>Order of Authors:</td>
<td>Bo Wang</td>
</tr>
<tr>
<td></td>
<td>Inhi Kim</td>
</tr>
<tr>
<td></td>
<td>Hai L. Vu</td>
</tr>
</tbody>
</table>
Short-Term Traffic Prediction Using A Spatial-Temporal CNN Model With Transfer Learning

Bo Wang
Institute of Transport Studies, Department of Civil Engineering
Monash University, VIC 3168, Australia
Email: Bo.Wang1@monash.edu

Inhi Kim
Institute of Transport Studies, Department of Civil Engineering
Monash University, VIC 3168, Australia
Email: Inhi.Kim@monash.edu

Hai L. Vu
Institute of Transport Studies, Department of Civil Engineering
Monash University, VIC 3168, Australia
Email: Hai.Vu@monash.edu

Word Count: 4,239 words + 2 tables = 4,739 words

Submitted [August 1, 2019]
ABSTRACT
Advanced neural network models have recently achieved competitive results for short-term traffic prediction. However, two problems require further study: 1. The effects of the different structured convolutional neural networks (CNN); and 2. The possibility of transfer learning techniques on traffic prediction problems. The unsuitable model structure could lead to inappropriate feature learning, and insufficient data could cause training difficulties. This paper investigates the influence of different CNN structures on a traffic model and how fine-tuning of the model affects the outcomes. The results show that a well-designed CNN structure can learn better traffic features, and fine-tuning the model benefits the situation which has little training data. This paper provides a positive reference for the structural design of CNN and verifies the effectiveness of the fine-tuning method on short-term traffic forecasting tasks.

Keywords: Short-term Traffic Prediction, Convolutional Neural Networks, Model Fine-tuning, Transfer Learning
INTRODUCTION

The research community and the transport industry have paid great attention to short-term traffic prediction. With different purposes of research or application, different traffic states (speed, volume, delay, travel time, and other factors) are forecasted. Short-term traffic prediction is a data-driven subject which uses historical data features to produce a model and predicts traffic in the near future. In the past two decades, the focus of this area has been transferred from statistical methods to neural networks (1). In particular, models based on deep learning structures have recently achieved a series of outstanding results for short-term traffic prediction (2).

Two types of model structures appear for innovative spatial-temporal deep learning models: Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). CNN is a type of feedforward neural network involving convolutional computations. The study of CNN began in the 1980s, and the theory of biological visual perception gave the inspiration for the structural design. The layering mechanism and convolutional kernels use fewer computations to learn grid-like topology features. In contrast, RNN is a type of neural network which calculates the output by sequentially feeding inputs through recurrent cells, and they have been used for problems related to sequential data. Long Short-Term Memory networks (LSTM) and Gated Recurrent Unit (GRU) are representative structures of the modern RNN. With the rapid development of computing power, CNN and RNN have been applied in computer vision, natural language processing, medical solutions processing, and transport studies (3).

For the short-term traffic prediction problem, many existing models were developed on top of the CNN and RNN models with modifications and combinations. For the CNN family, shallow and deep types of structure were derived. The shallow model consists of fewer CNN layers, which can predict traffic speed and flow (4, 5) while the deeper CNN models are created for traffic forecasting (6, 7) with the introduction of residual neural networks (8). On the other hand, RNN in this area can be summarized into two types of models: RNN layers combined models and structural modified models. The former attempts different ways to combine RNN layers (9-11), while the latter improves the model by introducing concepts to RNN cells (12, 13). Furthermore, the structural joining of CNN and RNN appears to be more effective for learning both spatial and temporal features (14-16).

However, there are two problems that cannot be neglected for short-term traffic prediction neural networks:

1. **Among the CNN models, it is not known what kind of convolutional kernel is effective especially when learning the spatial and temporal features of traffic.**

   Traffic data is inherently related to time and space; therefore, many models focus on learning spatial and temporal features for better forecasting performance. The design of data shape on the model inputs and convolutional kernels directly affects the output feature maps. However, most existing models use arbitrary kernel size (like $3 \times 3$) or try to learn traffic features through the outputs from hidden layers (16), which lacks physical meanings in the calculation.

2. **Transfer learning can improve supervised learning models to some extent, but the impact of this method on traffic prediction problems is rarely investigated.**

   Transfer learning is a research topic in machine learning, which applies the knowledge learned from solving a problem to another different but associated problem (17). The study of transfer learning is widespread in computer vision and natural language processing areas (18). Fine-tuning is one approach in transfer learning, which supports the initialization of a new model with trained weights and biases from another model. In practice, the systematic replacement or
expansion in traffic-related systems usually needs new sensors, which lack historical data for the model training. In this case, learned knowledge from the longer used sensors may help the forecasting accuracy for newly installed sensors by the application of the fine-tuning technique. However, detailed research on model fine-tuning is still unknown for short-term traffic prediction.

Therefore, the main contributions of this paper are:
1. A parallel CNN model which learns features from both spatial and temporal kernels is proposed and compared comprehensively with sequential CNN structure.
2. The performance of the fine-tuning method is verified. A model of the case study with limited data length is improved by initializing the weights and biases from a well-trained model.

METHODOLOGY

Data Description
Caltrans Performance Measurement System (PeMS) is the data source, which stores data from over 39,000 individual traffic detectors in San Diego, California, US. Three variables such as traffic speed, volume, and density are used for input data in this study. Two datasets ‘Modeling’ and ‘Case Study’ were created with different data length and location for further experiments. The Modeling dataset had three months of data for training, whereas the Case Study dataset contained only one-week of data. Figure 1 (a) shows the location of these two datasets on the northbound I5 highway, where equipped with loop detectors or radar detectors. The total length of the Modeling and Case Study highway links is 7.1 km and 8.9 km, respectively, and the average spacing between the detectors is 0.67 km.

Table 1 shows detailed information about these two datasets. Each dataset consists of 12 detectors, which are described as Postmile (a method for numbering the location of detectors in PeMS). The minimal time interval (one time-step) is 5 mins, and there is a total of 27,648 sequential time-steps for the Modeling and 2,016 sequential time-steps for the Case Study. For the model training and evaluation, each dataset was divided into training, validation and test sets. The training and validation subsets were used for K-fold cross-validation. Before each training epoch, these two subsets were randomly and discretely selected in 90% and 10% ratio with no overlap. The test set was not involved in any training process, and it was only used for model evaluation. Note that the specific index intervals of the three subsets also are listed in Table 1.
TABLE 1 Dataset Description

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Modeling</th>
<th>Case Study</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of locations</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>Traffic direction</td>
<td>Northbound</td>
<td>Northbound</td>
</tr>
<tr>
<td>Postmile</td>
<td>[34.033, 41.922]</td>
<td>[43.191, 51.327]</td>
</tr>
<tr>
<td>Time Interval</td>
<td>5 mins</td>
<td>5 mins</td>
</tr>
<tr>
<td>Start</td>
<td>1/03/2018 0:00</td>
<td>1/03/2018 0:00</td>
</tr>
<tr>
<td>End</td>
<td>31/05/2018 23:55</td>
<td>7/03/2018 23:55</td>
</tr>
<tr>
<td>Length (Time-steps)</td>
<td>27,648</td>
<td>2,016</td>
</tr>
<tr>
<td>Test Set</td>
<td>300 time-steps, [27,348, 27,648]</td>
<td>300 time-steps, [1,716, 2,016]</td>
</tr>
<tr>
<td>Training Set</td>
<td>Random 90% of [1, 27,348]</td>
<td>Random 90% of [1, 1,716]</td>
</tr>
<tr>
<td>Validation Set</td>
<td>Random 10% of [1, 27,348]</td>
<td>Random 10% of [1, 1,716]</td>
</tr>
</tbody>
</table>

The data shape of the model inputs and outputs are different. Aggregated data were used every 5 mins. For each time-step, all 12 detectors provided three traffic variables: density, volume, and speed. The input and output data can be denoted as:

Input Data Shape = \([N, D, T_{in}, A_{in}]\) \hspace{1cm} (1)

Output Data Shape = \([N, D, T_{out}, A_{out}]\) \hspace{1cm} (2)

where \(N\) is batch size, \(D\) is the number of detectors, \(T_{in}\) and \(T_{out}\) are input and output time-steps (i.e. 5 mins), and \(A_{in}\) and \(A_{out}\) are the number of input and output attributes, respectively. The forecasting model uses 24 hours of data (\(T_{in} = 288\)) to predict the future traffic speed in the next several sequential time-steps; Figure 1 (b) demonstrates the structure of the input data. As shown in the figure, the input tensor consists of three channels, i.e. \(A_{in} = 3\) (speed, density, and flow). For the output tensor, it only has one channel, i.e. \(A_{out} = 1\) (speed).

Figure 2 displays a typical input sample from the test set in Modeling dataset, which was recorded from 09:00 to 09:05 AM on May 31, 2018, from 12 detectors. The four subplots have similar data distribution but are colored to represent different perspectives (location, density, flow, and speed). The same plotting layout following the fundamental diagram (FD) in traffic flow theory was used where each subplot consists of three inner relations of traffic variables. Subplots (a) show that all detectors generally follow the ideal relationship suggested by FD, although they have slightly different data distributions. Subplots (b), (c) and (d) present the three 2D graphs separately that show three attributes (density, flow, and speed) in color gradation. All graphs here show each attribute changes gradually in the relation of two other attributes. Figure 2 shows that mutual constraints exist on the internal attributes of the input data. Therefore, the model should be able to learn the relationship between three historical traffic factors and future speed from the training process.
Figure 2 A Sample of Input Data with different perspectives
As data pre-processing is essential for deep learning models, data normalization and standardization for traffic speed, density, and volume were conducted. Data normalization maps the raw data into a new range ([0, 1] is always by convention), and it helps the model to converge faster and achieve higher accuracy. Data standardization rescales the data with a mean ($\mu$) of 0 and a standard deviation ($\sigma$) of 1. This procedure makes the different variables comparable in a new distribution. The equations for data normalization and standardization can be expressed as:

\[
Y = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \quad (3)
\]

\[
Y = \frac{X - \mu}{\sigma} \quad (4)
\]

where $X$ is the raw data, $X_{\text{min}}$ and $X_{\text{max}}$ are the minima and maximum value of $X$, and $\mu$ and $\sigma$ are the mean value and standard deviation of $X$, respectively.

**Forecasting Model**

CNN plays a significant role in many areas, and all related works are built on a one core idea: using convolution to capture the features from input data. Assume $X$ is an $m \times m$ input matrix and $K$ is an $n \times n$ filter matrix; the 2D convolution can be given as:

\[
z(u, v) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} X_{i+u, j+v} \cdot K_{i, j} \cdot X(i, j) + B \quad (5)
\]

\[
X(i, j) = \begin{cases} 1, & 0 \leq i, j \leq n \\ 0, & \text{others} \end{cases} \quad (6)
\]

where $i, j \in [0, m]$, $u, v \in [0, n]$, $B$ is the biases tensor, the $X(i, j)$ function ensures the convolution only happens in valid areas, and $z$ is the feature map. The feature map often is forward of an activation function to enable non-linear modeling:

\[
a_{l,u,v}^l = f(z_{l,u,v}) \quad (7)
\]

where $l$ is the $l$-th layer, $f$ is an activate function, and $a$ is the corresponding non-linear output. Equation (5) is the complete forward convolution progress in neural networks, which uses a $n \times n$ filter to extract the features from a $m \times m$ input data. Note that the input data here can be more than one channel, and the filter would perform with the same movement but through all channels. In this paper, CNN is used mainly to capture traffic features and make speed predictions.

Figure 3 (a) shows the structure of the proposed forecasting model, which uses a structure of two parallel CNNs to capture the temporal and spatial features. For conciseness, the batch size of model inputs and outputs is not included in Figure 3 (a). The main difference between these two CNNs is the filter size, which is a trainable tensor (i.e. weight and bias) for element-wise multiplication in convolutions. As shown in subplots (b) and (c), the filter sizes of temporal (Conv 1) and spatial CNN layer (Conv 2) are $(1 \times 12 \times 3)$ and $(288 \times 1 \times 3)$, respectively. The first type of filter moves through 288 time-steps whereas the second type calculates convolution result over 12 locations. Furthermore, the non-linear function, rectified linear unit (ReLU), is performed after two CNN layers. Finally, the linear layer concatenates all outputs from the previous two layers, then performs linear regression for correct output size.
Figure 3 Model Structure and two convolutional approaches

Note that the data shape of the output in (a) is an example of one time-step output, but it is flexible to change for a sequential prediction. This paper sets a group of output duration as \{1, 2, 3, 4\} for comparison, i.e. forecasting the traffic speed in next \{5, 10, 15, 20\} mins. Moreover, the output size of the CNN layers is a hyperparameter, and the higher value could make the model more complex. The output size was set as four for both two CNN layers; accordingly, there are eight trainable filters.

Performance Metrics

To measure the model performance, mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE) are used for performance metrics. They are given as:

\[
MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2
\]  

\[
RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}
\]  

\[
MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|
\]  

\[
MAPE = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right|
\]  

where \(N\) is the number of the sample, \(y\) is the real data and \(\hat{y}\) is the forecasting value.
Training and Evaluating Workflow

Figure 4 shows the workflow of the model training and evaluation for the two datasets. The three model sections in this figure denote the training processes. They have the same model structure but use two different methods to initialize the weights and biases in the hidden layers. The first method (Random) adopts random numbers from a uniform distribution for initialization \((19)\). The model continues optimizing these values through backpropagation process. The other method (fine-tuning) in the Case Study is called model fine-tuning, which initializes weights and biases from a trained model (Modeling).

![Figure 4 The workflow of experiments](image)

The experiments begin with the Modeling dataset, and the three subsets are used for model training and evaluation (trained weights and biases and model performance gained). After the experiment on the Modeling dataset, two initialization methods are performed on the Case Study dataset before the training and evaluation processes.

RESULTS

All experiments were supported by M3 (a high-performance computing cluster provided by Monash University). For each training container, 2 CPUs, 1 Tesla P100 GPU, and 8GB RAM were used. The Adam algorithm with a learning rate of \(10^{-3}\) was used for optimization. The batch size and training epoch of all experiments was set to 32 and 100, respectively.

Two groups of experiments: the control group and the experimental group were set up. The experimental group used the proposed model structure (parallel CNNs). A sequential ‘CNN+LSTM’ model was the control group implemented to validate the proposed model structure. It consists of a CNN layer, a linear layer, and an LSTM layer. The control group model learns features layer by layer, which causes the following LSTM layer to be trained by the hidden output from the CNN layer only. However, the proposed parallel structure in the experimental group learns spatial and temporal features simultaneously. All groups are trained for forecasting the future traffic speed with several sequential time-steps.
Table 2 shows the detailed results of the experiments. Each group performs model training and evaluation using two different datasets. The data duration of the Modeling and Case Study models are three months and one week, respectively. Moreover, there are two different approaches to initialize the model weights and biases: random and fine-tuning. We used a trained model state from the Modeling dataset (A) to fine-tune the model in the Case Study (B). The different hidden size of the middle layer would lead to different model performance. For a fair comparison, the number of model parameters also is listed. A higher number of parameters represent a more complex model.

The detailed evaluation results in Table 2 indicate that:

- The results in the experimental group used fewer parameters and produced better results than the control group, which indicates that the proposed two parallel CNNs outperformed the CNN-LSTM structure.
- The corresponding results in the Modeling dataset are better than in the Case Study dataset, which means that more training data result in a better performance.
- With the same random initialization, fewer training data lead to worse performance in both control and experimental groups.
- The trained model can be used for other locations (from A to B), and the fine-tuning results of the Case Study are improved with limited training data. The average improvement of the control group and experimental group is 7.4% and 39.6%, respectively (based on the MSE).
DISCUSSION

Fine-tuning Method

To better understand the benefits of the fine-tuning model, Figure 5 plots MSEs of the randomly initialized model and the fine-tuned model for the first 20 training epochs. There are four subplots in Figure 5 for four different output durations: {5, 10, 15, 20} mins. In each subplot, the x-axis represents the training epoch, and the y-axis represents the MSE of the corresponding validation set. The fine-tuning weights are adaptive for all output durations, and they help the model converge much faster than when random starting values (see all subplots) are used. Note that the tiny fluctuations of MSE values exist in all subplots, and this may be caused by the randomness of the initialization of the training and validation subsets. Note that the results in Table 2 show that the fine-tuning model could achieve far better performance with limited or less training data after full training epochs.

The supervised learning models require a mass of historical data for training. At the same time, there are many reconstructions or new installations of traffic facilities that lack enough historical data for the forecasting model. The fine-tuning method developed in this paper can tackle this problem by introducing the trained model status from other similar locations. This paper only tested this method on a limited data duration (3 months) and road segments (12 detectors). More complicated experimental situations for transfer learning in transport is to be investigated in our future work.

CNN Structure

The control group uses a sequential structure of CNN+LSTM. As the number of model parameters is at the scale of $10^6$; many hidden features can be created inside the model. However, there exist two problems with this structure: 1. The CNN layer uses a $3 \times 3$ kernel size for convolution, with learning features from a 3 locations $\times$ 3 time $-$ steps $\times$ 3 attributes tensor. However, the forecasting output is a 12 locations $\times$ N future time $-$ steps $\times$ 1 attribute tensor. This approach produces more feature maps than the experimental group, but spatial and temporal features were split into separate domains. 2. The LSTM learns temporal features from sequential inputs, but the inputs are hidden outputs from the CNN layer rather than inputs from traffic data.

The experimental group uses two parallel CNN layers. They help the model extract spatial and temporal features directly from traffic data, thereby avoiding the feature learning problems occurring in sequence models. Although it holds fewer parameters than the control group, this structure is more efficient in forecasting results. However, the results in Table 2 are
not comprehensive, which is due to the limited sample size and the hyperparameter setting. More experiments on different input and CNN parameters should be studied in future work.

Performance Over Time

Table 2 only shows the average errors on the whole test set (300 time-steps). It is difficult to observe the error change over different times. Therefore, Figures 6 (a) and (b) demonstrate the RMSE and real traffic speed distribution of the test set in Modeling model. The 12 locations are marked by 12 different colors showing different fluctuations over time.

Figure 6 The RMSE and speed distribution in the test set

According to Table 2, the average RMSE value of subplot (a) is 1.518. However, the error distribution varies significantly with different hours. We note two issues of RMSE preventing the model from further improvement: 1. The location 39.793 around 10:30 AM; 2. The general fluctuation between 2:00 and 7:00 PM. At the same time, the corresponding speed in subplot (b) also changes largely. The unstable speed values may be caused by increasing traffic volume, bad weather, accidents or other external factors. Consequently, the proposed model has difficulty forecasting drastic changing speed. In future studies, the influence of external factors can also be considered.
CONCLUSIONS

This paper studied the short-term traffic speed prediction problem by using neural network models. A novel parallel CNN structure for solving the feature learning problem in sequential models was proposed. The results showed that the proposed model could learn traffic features and provide a better prediction than the sequential model structure. Furthermore, the fine-tuning method enabled the model of a neural network to converge more quickly with better forecasting results when limited training data are available in the traffic prediction scenarios.

In the future, more complex input and output data will be considered. Since traffic flow, density, and speed are essential factors in traffic flow theory, further studies will aim to integrate more domain knowledge to guide the neural network training.

ACKNOWLEDGMENTS

Accessing to the high-performance computing system at Monash (Massive 3 or M3) to process and analyze the data for this study is gratefully acknowledged.

AUTHOR CONTRIBUTIONS

The authors confirm contribution to the paper as follows. Study conception and design: B. Wang, I. Kim, and H. Vu; data collection: B. Wang; analysis and interpretation of results: B. Wang, I. Kim, and H. Vu; draft manuscript preparation: B. Wang, I. Kim, and H. Vu. All authors reviewed the results and approved the final version of the manuscript.
REFERENCES


