# Short-Term Forecasting of Traffic Flow and Speed: A Deep Learning Approach

Lingxiao Zhou[1] and Xiqun (Michael) Chen[2]

[1]Institute of Transportation Engineering, College of Civil Engineering and Architecture, Zhejiang Univ., 866 Yuhangtang Rd., Hangzhou 310058, China. E-mail: chenxiqun@zju.edu.cn
[2]Institute of Transportation Engineering, College of Civil Engineering and Architecture, Zhejiang Univ., 866 Yuhangtang Rd., Hangzhou 310058, China (corresponding author). E-mail: chenxiqun@zju.edu.cn

Downloaded from ascelibrary.org by Newcastle University on 03/28/19. Copyright ASCE. For personal use only; all rights reserved.

## ABSTRACT

Real-time short-term traffic state prediction is essential for traffic management and is often conducted, based on historical data, at different traffic detection locations. However, the spatial and temporal dependencies of different detectors and different traffic state indicators need to be considered simultaneously, which makes short-term traffic prediction challenging. In this paper, we propose a deep learning based approach to simultaneously predict multi-step ahead traffic flow and speed for multiple detectors on a freeway. Based on the data collected by multiple loop detectors on the freeway, this paper transforms the time-space tensor into an image. The convolutional neural network (CNN) is employed to handle the multi-input multi-output problem. Compared to the support vector regression (SVR) and historical average (HA) models, the CNN-based model performs higher prediction accuracy and has a better performance in multistep traffic prediction. The proposed method can predict traffic speed and flow of several future time intervals within one prediction step, which saves a lot of prediction time.

**KEYWORDS:** Traffic flow and speed prediction; deep learning; convolutional neural network (CNN); support vector regression (SVR); historical average (HA)

## 1 INTRODUCTION

Accurate and timely short-term forecasting of the near-future traffic state is the basis of proactive traffic control, real-time information provision for travelers, dynamic vehicle routing, etc., which are the vital components of intelligent transportation systems (ITS). Among the numerous traffic state indicators, traffic speed and flow are among the most important ones. A lot of fixed traffic detectors, e.g., road cameras, inductive loop detectors (ILD), and microwave sensors, can collect real-time information on traffic speed and vehicle counts. Most of the detectors are installed discretely and sparsely. However, understanding traffic speed and flow for the whole road segment, or even network, rather than on one discrete location is more important and meaningful to provide travelers with complete information and then assist them on making better route choices. Meanwhile, multi-detector speed and flow along the whole road segment can support operators in advancing traffic management.

However, when we come to multi-detector multi-step short-term traffic prediction, some challenges will arise on the model establishment. For example, efficient models should be able to deal with high dimensions and a large amount of data, which requires more costly computational processes. The spatial-temporal correlation must be identified by new models, which require more intelligent and complex model structures. Unfortunately, traditional traffic prediction models often treat speed and flow as single and independent time-series data, which ignore the spatial correlation. Most of the current models predict the speed and flow separately. Thus, existing models may be difficult to simultaneously predict multi-detector multi-step speed and

© ASCE

flow.

Numerous models and algorithms have been proposed to predict traffic states. Davis et al. (1991) and Xia et al. (2016) used *k*-nearest neighbors (KNN) to predict traffic speed and volume. Wu et al. (2004) applied the support vector regression (SVR) to predict travel time. Then, SVR was improved by Castro-Neto et al. (2009) and Hong (2011). More recently, the random forest (RF) and gradient boosting regression trees (GBRT) were also used in traffic flow and travel time prediction (Zhang and Haghani, 2015). However, these models ignore the spatial-temporal features of the traffic, so they cannot deal with multi-detector prediction effectively.

In the literature, there have been some studies which predicted traffic states by considering temporal and spatial correlations of neighboring traffic sensors. For example, Li et al. (2014) proposed a multi-model ensemble method to deal with data from three sensors. In addition, Li et al. (2016) proposed a hybrid strategy to handle the prediction problem by considering both temporal and spatial features. Min and Wynter (2011) developed a method for traffic prediction considering the spatial characteristics of a road network.

With the emergence of big data, deep learning (DL) methods have developed rapidly in recent years; lots of DL models were applied to predict traffic states. With the help of DL, the temporal and spatial correlations of traffic states measured by different sensors can be extracted in a better way. Ma et al. (2017) proposed a convolutional neural network (CNN) based method to learn the traffic speed as images. Ke et al. (2017) proposed a new DL approach, named the fusion convolutional long short-term memory network to capture the spatiotemporal characteristics and correlations of explanatory variables. Polson and Sokolov (2017) proposed a deep learning model that combined a linear model and a sequence of tanh layers to predict traffic flow. A stacked auto encoder was applied to the deep architecture to predict traffic flow (Lv et al. 2015). Zhang et al. (2017) used a residual CNN to predict crowd flow. However, these studies mainly focus on the prediction of a single traffic state indicator.

To fill the gap that various traffic state indicators of multiple detectors cannot be predicted together, this paper proposes a CNN-based model to simultaneously predict multi-detector multi-step speed and flow. The SVR and historical average (HA) models were built as a comparison. The contributions of this paper are twofold: (1) We build a multi-input-multi-output CNN model, in which traffic speed and flow of all detectors in several time intervals on a road are predicted simultaneously in one model; (2) Temporal and spatial correlations are taken into account to improve the prediction accuracy. The influences of upstream and downstream sensors in different time periods are analyzed.

The rest of this paper is organized as follows: Section 2 presents the method to convert traffic data into 3-dimension tensors, and CNN for multi-detector speed and flow prediction that considers temporal and spatial correlations. In Section 3, numerical tests are proposed using real-world traffic detection data from the freeway Performance Measurement System (PeMS) in California. Section 4 draws conclusions and outlooks the further research directions.

## 2 METHODOLOGY

To predict multi-detector speed and flow, both temporal and spatial correlations should be simultaneously considered to construct the comprehensive model input. We present a 3-D tensor, the *x*-dimension, and *y*-dimension of which represent the temporal and spatial information, respectively. The *z*-dimension has two indexes, i.e., traffic speed and flow. The elements in the tensor are the values of traffic speed and flow at different time stamps and locations. The tensor can be regarded as an image with two channels, in which every pixel corresponds to the value in

Downloaded from ascelibrary.org by Newcastle University on 03/28/19. Copyright ASCE. For personal use only; all rights reserved.

© ASCE

the tensor. That is, the image is of *M*-pixel height, *N*-pixel width, and *D*-pixel depth, where *M*, *N*, and *D* are the three dimensions of the 3-D tensor. In this paper, the value of *D* is 2 (i.e., speed and flow). CNN is a powerful and efficient method in image processing and has been applied to image recognition. We propose an approach that converts the multi-detector traffic speed and flow information into images and then adds them into a CNN model as inputs for traffic prediction.

### 2.1 Converting the Traffic Information into Images

There are many sensors installed at different locations along the freeway for traffic state detection. Through the information of all the sensors, the spatiotemporal traffic information at different locations can be estimated and converted into a tensor that corresponds to an image.

In the *x*-dimension (time dimension), time intervals usually depend on the sampling frequency of the sensors and often range from less than 1s (e.g., radar sensors) to 5 min (e.g., loop detectors). However, narrow intervals, such as 5s, are meaningless in traffic prediction. It is useful to obtain a temperate time interval, e.g., 5 min, according to the field experience. In the *y*-dimension (space dimension), when the sensors are equipped on the road, the spatial locations of them are fixed. When we choose the upstream end of the road as a reference point, the sensors can be sorted by their driving distance from the reference point. In the *z*-dimension, two indexes are created, i.e., speed and flow. Finally, the tensor can be given as:

$$\mathcal{M} = [\boldsymbol{S}, \boldsymbol{F}] \qquad \boldsymbol{S} = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1N} \\ s_{21} & s_{22} & \cdots & s_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ s_{M1} & s_{M2} & \cdots & s_{MN} \end{bmatrix} \qquad \boldsymbol{F} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1N} \\ f_{21} & f_{22} & \cdots & f_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ f_{M1} & f_{M2} & \cdots & f_{MN} \end{bmatrix} \tag{1}$$

where $\boldsymbol{S}$ represents the speed time-space matrix, $\boldsymbol{F}$ represents the flow time-space matrix; *N* is the length of time intervals, *M* is the number of different sensor locations; the *i*th row vector of $\boldsymbol{S}$ and $\boldsymbol{F}$ is the traffic speed and flow at the *i*th location of the road at different time stamps, respectively; the *j*th column vector of $\boldsymbol{S}$ and $\boldsymbol{F}$ is the traffic speed and flow of different sensor locations at the *j*th time interval, respectively; and $s_{ij}, f_{ij}$ represent the traffic speed and traffic flow at detector location *i* at time *j*. Tensor $\mathcal{M}$ with the size of $M \times N \times 2$ can be regarded as an image with 2 channels. Eq. 1 can also be written as:

$$\mathcal{M} = \begin{bmatrix} \boldsymbol{m}_{11} & \boldsymbol{m}_{12} & \dots & \boldsymbol{m}_{1N} \\ \boldsymbol{m}_{21} & \boldsymbol{m}_{22} & \dots & \boldsymbol{m}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{m}_{M1} & \boldsymbol{m}_{M2} & \dots & \boldsymbol{m}_{MN} \end{bmatrix} \qquad \boldsymbol{m}_{ij} = \begin{bmatrix} s_{ij}, j_{ij} \end{bmatrix} \tag{2}$$

where $\boldsymbol{m}_{ij}$ is the vector including traffic speed and flow at detector location *i* at time *j*.

### 2.2 CNN for Traffic Prediction

Figure 1 shows the structure of the CNN model for traffic prediction. The model contains four main parts: model input, feature extraction, fully-connected layer, and model output.

As illustrated in Section 2.1, the model input is the image obtained by constructing the speed-flow time-space tensor. Let the input and output time intervals be *I* and *O* (that is, we use *I* time intervals' history information to predict the *O* time intervals' future condition), respectively. The total number of time intervals is *T*.

Downloaded from ascelibrary.org by Newcastle University on 03/28/19. Copyright ASCE. For personal use only; all rights reserved.

© ASCE

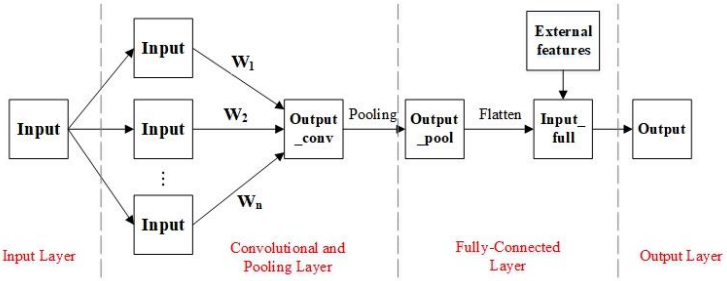



**Figure 1. Framework of CNN-based traffic prediction model**

So that the model input and output can be written as:

$$\boldsymbol{x}^i = \left[\boldsymbol{m}_i, \boldsymbol{m}_{i+1}, \cdots, \boldsymbol{m}_{i+I-1}\right], \quad i \in \left[1, T-I-O+1\right] \tag{3}$$

$$\boldsymbol{y}^i = \left[\boldsymbol{m}_{I+i}, \boldsymbol{m}_{I+i+1}, \cdots, \boldsymbol{m}_{I+i+O-1}\right], \quad i \in \left[1, T-I-O+1\right] \tag{4}$$

where $i$ is the $i$th sample, $\boldsymbol{m}_i$ is the $i$th column vector in tensor $\mathcal{M}$.

The model input is handled by several convolutional layers and pooling layers. When we put a tensor $\mathcal{X}$ with width $M$, height $N$, and depth $D$, a convolutional filter $\boldsymbol{W}$ with dimensions of $m \times n$ ($m \le M$, $n \le N$) is applied to calculate the output as:

$$\boldsymbol{Y} = \boldsymbol{W} * \mathcal{X} = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,N-n+1} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,N-n+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{M-m+1,1} & y_{M-m+1,2} & \cdots & y_{M-m+1,N-n+1} \end{bmatrix} \tag{5}$$

$$y_{ab} = \sum_{d=1}^{D} \sum_{i=1}^{m} \sum_{j=1}^{n} \left( w_{ij} \cdot x_{i+a-1, j+b-1, d} \right), \quad a \in \left[1, M-m+1\right], b \in \left[1, N-n+1\right] \tag{6}$$

where we use the symbol * to represent the convolution operation; $D$, $M$, and $N$ are 3 dimensions of the input tensor $\mathcal{X}$; $m$, $n$ are the 2 dimensions of the filter $\boldsymbol{W}$; $x_{i+a-1,j+b-1,d}$ is the value of $\mathcal{X}$ at position $i+a-1$, $j+b-1$ in the $d$th channel, and $w_{ij}$ is the coefficient of the filter at position $i$ and $j$. $\boldsymbol{Y}$ is the output matrix.

Note that in the convolution operation, all the channels share the same filters, and the result of each channel is summed up. In this paper, we have $K$ filters in one convolutional layer, that is, repeat the convolution operation for $K$ times and stack the $K$ matrices in the third dimension, so the output $\boldsymbol{Y}$ becomes a tensor with $K$ channels (depth). The specific value of $K$ will be shown in Section 3.3.

Each convolutional layer is followed by a pooling layer designed to downsample and aggregate data so that the model is more efficient. In this paper, we apply the max pooling method, which is illustrated as:

Dividing $\boldsymbol{Y}$ into several blocks without overlapping, each block has the size of $\lambda \times \tau$, where $\lambda$ and $\tau$ are the width and height of the pooling filter, respectively. So the $(ij)$th block can be denoted by $\boldsymbol{Y}_{ij}$, and the max pooling result can be written as:

$$\boldsymbol{Z} = \text{pool}(\boldsymbol{Y}) = \begin{bmatrix} \max(\boldsymbol{Y}_{11}) & \max(\boldsymbol{Y}_{12}) & \cdots & \max(\boldsymbol{Y}_{1j}) \\ \max(\boldsymbol{Y}_{21}) & \max(\boldsymbol{Y}_{22}) & \cdots & \max(\boldsymbol{Y}_{2j}) \\ \vdots & \vdots & \ddots & \vdots \\ \max(\boldsymbol{Y}_{i1}) & \max(\boldsymbol{Y}_{i2}) & \cdots & \max(\boldsymbol{Y}_{ij}) \end{bmatrix} \tag{7}$$

where $\max(\cdot)$ represents the maximum value of $\boldsymbol{Y}_{ij}$; $\boldsymbol{Z}$ is the output.

Downloaded from ascelibrary.org by Newcastle University on 03/28/19. Copyright ASCE. For personal use only; all rights reserved.

© ASCE

After one pooling operation, the height and width of input are reduced into $1/\lambda$ and $1/\tau$ of the original size, respectively. Note that each channel does not sum their pooling result together. So the depth is not changed. That is, if the convolutional layer's output $\boldsymbol{Y}$ is a tensor with $K$ channels, the output of the pooling layer will still have $K$ channels. If we denote our model input, output, and parameters of the $l$th layers as $\boldsymbol{x}_l$, $\boldsymbol{o}_l$, ($\boldsymbol{W}_l$, $\boldsymbol{b}_l$), respectively. The output in the $l$th convolutional-pooling layer can be written as:

$$\boldsymbol{o}_l = \text{pool}\left(\sigma\left(\boldsymbol{W}_l * \boldsymbol{x}_l + \boldsymbol{b}_l\right)\right) \tag{8}$$

where $\sigma$ is the activation function.

In this paper, we choose two activation functions: the tanh function for convolutional layers and the sigmoid function for the fully connected layer. The two functions are given by:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{9}$$

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{10}$$

After the convolutional layers, we flatten the output into one-dimension vector. Then we divide a day into 288 pieces (each piece corresponds to 5-min time interval) and then add the time of day and day of week into the flattened vector as additional features. The vector is transformed into model outputs:

$$\hat{\boldsymbol{y}} = \boldsymbol{W}_f \boldsymbol{o}_L^{\text{flatten}} + \boldsymbol{b}_f \tag{11}$$

where $\boldsymbol{W}_f$ and $\boldsymbol{b}_f$ are the parameters of the fully connected layer; $\hat{\boldsymbol{y}}$ is the predicted traffic speed and flow; $\boldsymbol{o}_L^{\text{flatten}}$ is the output of the last convolutional layers (the $L$th), which is flattened into one dimension in advance.

Since the traffic speed and flow data have different magnitudes, we normalize all the input data with the method:

$$x = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{12}$$

where $x$ is the value of speed or flow; $x_{\max}$ and $x_{\min}$ are the maximum and minimum of speed or flow, respectively.

## 3 NUMERICAL EXPERIMENTS

### 3.1 Data Description

To test the performance of the multi-detector simultaneous prediction method, we use the loop detector data collected from 57 vehicle detection stations along Freeway I405, northbound, Los Angeles, California, US. These 57 stations are randomly placed along the road. The distance between the nearest 2 stations is 0.095 mile and the furthest is 0.66 mile. The data used in this paper were extracted from PeMS. Figure 2 shows the location of the study site. Each of the 57 vehicle detection stations includes several loop detectors that measure passing vehicles and speeds lane by lane for every 30s. The raw data were aggregated into 5-min time intervals and all lanes' data were aggregated into one section. The 5-min mean speed and flow collected between June 1, 2017 (Thursday), and June 28, 2017 (Wednesday). The first 3-weeks of data were used as the training set and the remaining 1-week of data were used as the test set.

Two prediction experiments are designed to test the CNN-based method in predicting the

© ASCE



Downloaded from ascelibrary.org by Newcastle University on 03/28/19. Copyright ASCE. For personal use only; all rights reserved.

multi-detector traffic speed and flow. The main difference between these two experiments is the prediction time steps:

**Experiment I**: Predict 10-min (2 steps) traffic flow and speed using the previous 60-min traffic data (12 steps);

**Experiment II**: Predict 30-min (6 steps) traffic flow and speed using the previous 60-min traffic data (12 steps)

The performance of the CNN method and the comparison with other benchmark models in the two experiments will be presented in Section 3.4

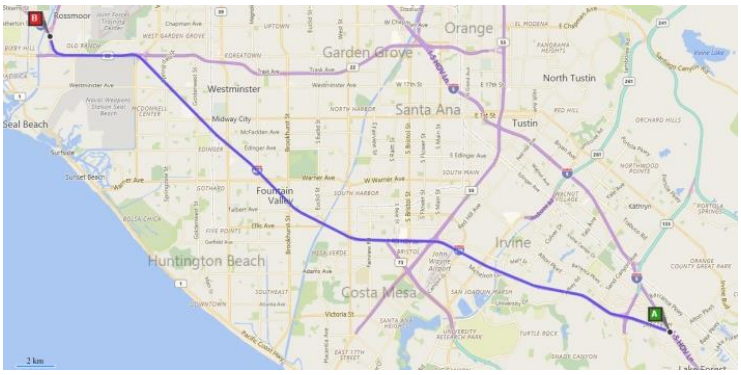



**Figure 2. Study site at I405-N, Los Angeles, CA.**

### 3.2 Measures of Effectiveness (MoE)

In this paper, we choose the following five MoE to measure prediction errors or goodness-of-fit of the CNN-based method, SVR, and HA. The root mean square error (RMSE), normalized root mean square error (NRMSE), symmetric mean absolute percentage error (SMAPE), and R-square ($R^2$) are defined by:

$$\mathrm{RMSE} = \sqrt{\frac{1}{MN}\sum_{t}^{N}\sum_{i}^{M}\left[y_{it} - \hat{y}_{it}\right]^2} \tag{13}$$

$$\mathrm{NRMSE} = \sqrt{\frac{\sum_{t}^{N}\sum_{i}^{M}\left[y_{it} - \hat{y}_{it}\right]^2}{\sum_{t}^{N}\sum_{i}^{M}\left(y_{it}\right)^2}} \times 100\% \tag{14}$$

$$\mathrm{SMAPE1} = \frac{1}{MN}\sum_{t}^{N}\sum_{i}^{M}\frac{\left|y_{it} - \hat{y}_{it}\right|}{y_{it} + \hat{y}_{it}} \times 100\% \tag{15}$$

$$\mathrm{SMAPE2} = \frac{\sum_{t}^{N}\sum_{i}^{M}\left|y_{it} - \hat{y}_{it}\right|}{\sum_{t}^{N}\sum_{i}^{M}\left[y_{it} + \hat{y}_{it}\right]} \times 100\% \tag{16}$$

$$\mathrm{R}^2 = 1 - \frac{\sum_{t}^{N}\sum_{i}^{M}\left[y_{it} - \hat{y}_{it}\right]^2}{\sum_{t}^{N}\sum_{i}^{M}\left[y_{it} - \bar{y}\right]^2} \tag{17}$$

Downloaded from ascelibrary.org by Newcastle University on 03/28/19. Copyright ASCE. For personal use only; all rights reserved.

© ASCE

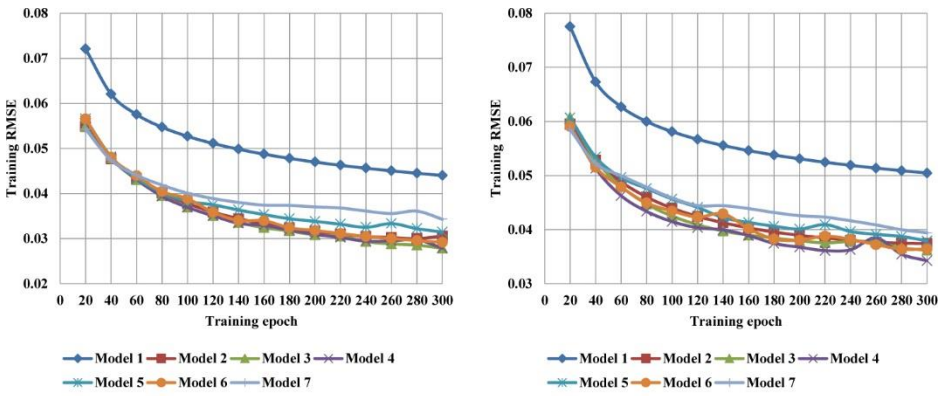
where $M$ is the number of detectors and $N$ is the number of time intervals; $y_{it}$ and $\hat{y}_{it}$ are true and predicted values of speed or flow, respectively; $\bar{y}$ is the mean of all the $y_{it}$.

### 3.3 Parameters Tuning

When we establish a CNN model, two main parameters should be considered for tuning: (1) The number of hidden layers of CNN, which is also known as the depth; and (2) parameters of the convolutional layers and pooling layers (e.g., the filter number, filter size, and pooling method). In this paper, we refer to two well-known examples: LeNet (LeCun et al. 1989) and AlexNet (Krizhevsky 2012), and choose the convolutional and pooling filter size as (3, 3) and (2, 2), respectively. The convolutional stride is 1. The max pooling method in the pooling layers is used. Since several parameter combinations (e.g., the depth and convolutional filter numbers) of CNN should be tested to find the optima, we choose 7 combinations which are shown in Table 1, in which each layer has one convolutional layer followed by one pooling layer. The numbers represent the quantities of convolutional filters in each layer. Note that Model-1 only has a fully-connected layer, which transforms input to output straightly.

We use RMSE as the loss function of the CNN model for parameters tuning. Figure 3 shows the results of training RMSE of different CNN models with the increase of the training epoch. In Experiment 1, when the training epoch is 200, the training RMSE has achieved a low degree. Though the RMSE continues to decrease as the epoch increases, the training time also increases. It's worthless to pursue a little performance improvement at the price of a large increase in time. So a reasonable epoch is chosen so that the RMSE reaches an acceptable degree with a temperate training time. Finally, we choose the training epoch as 200, and the best model is Model-3. As for Experiment 2, the best model is Model-4 with the training epoch 200.

We employ SVR for a comparison, with the parameters as $\varepsilon$=0.1 and $C$=1, and the kernel function is linear. The HA method is also used as a comparison; we use 12 time intervals of historical data to calculate the mean, which is then used as the prediction result. The prediction results will be shown in Section 3.4.




**(a) Experiment I** **(b) Experiment II**

**Figure 3. Training RMSE of different models.**

### 3.4 Results

The models were built on a PC with the configuration as follows: 3.6 GHz, 4 Cores, Inter (R) Core (TM) i7-7700HQ, 16 GB memory. Table 2 and Table 3 show the prediction results of

© ASCE



Downloaded from ascelibrary.org by Newcastle University on 03/28/19. Copyright ASCE. For personal use only; all rights reserved.

CNN, SVR, and HA used in Experiments 1 and 2, respectively. Note that when we establish CNN models, 3 models are built. The main one (we call it the best CNN model below) uses both flow and speed as input and output (that is, the image depth is 2). The other two models only use flow or speed (that is, the image depth is 1) separately. As for SVR and HA models, since it will be lack of model interpretation if we put speed and flow into one vector, only a vector of speed or flow is used as the input when training SVR and HA models. So, only 2 models with only flow input and only speed input were built. The 'flow & speed data' model was just built by combining the errors of the two separate models together.

**Table 1. Combinations of the Depth and Filter Number of CNN models**

| CNN model | Layer(s) | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| Model-1 | 0 | 0 | 0 |
| Model-2 | 32 | 0 | 0 |
| Model-3 | 64 | 0 | 0 |
| Model-4 | 128 | 0 | 0 |
| Model-5 | 64 | 32 | 0 |
| Model-6 | 128 | 64 | 0 |
| Model-7 | 128 | 64 | 32 |

**Table 2. Comparison of Prediction Performance in Experiment 1**

| Model | | MoE | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Training time (min) | Test time (s) | RMSE | NRMSE (%) | SMAPE1 (%) | SMAPE2 (%) | $R^2$ |
| CNN | Flow & speed data | 277.13 | 2.0 | 0.036 | 5.754 | 3.379 | 2.135 | 0.978 |
| | *Flow error* | NA | NA | 0.036 | 7.386 | 4.758 | 3.048 | 0.976 |
| | *Speed error* | NA | NA | 0.036 | 4.877 | 1.999 | 1.596 | 0.939 |
| | Flow data only | 273.86 | 1.9 | 0.036 | 7.472 | 5.643 | 3.196 | 0.976 |
| | Speed data only | 275.77 | 1.9 | 0.034 | 4.692 | 1.951 | 1.531 | 0.944 |
| SVR | Flow & speed data | 219.18 | 31.5 | 0.043 | 6.877 | 5.770 | 3.027 | 0.969 |
| | Flow data only | 108.54 | 16.1 | 0.044 | 9.129 | 8.744 | 4.099 | 0.964 |
| | Speed data only | 110.64 | 15.4 | 0.041 | 5.624 | 2.796 | 2.375 | 0.919 |
| HA | Flow & speed data | NA | 6.4 | 0.060 | 9.628 | 5.407 | 3.262 | 0.939 |
| | Flow data only | NA | 3.2 | 0.059 | 12.259 | 7.997 | 5.094 | 0.935 |
| | Speed data only | NA | 3.2 | 0.060 | 8.226 | 2.818 | 2.182 | 0.826 |

NA: not applicable.

Table 2 and Table 3 show that the CNN models outperform the HA models in both experiments. In comparison, we can see that the CNN models using both flow and speed as multiple inputs perform a little bit worse than using only flow or speed data; our loss function is the total RMSE when tuning parameters in Section 3.2. When the loss function considers flow RMSE and speed RMSE separately, the result should be better. However, since we build multi-input multi-output models, traffic flow and speed can be predicted at one time, saving computational time in enabling real-time applications. The training time reduces by half with a tiny decrease in the prediction accuracy, which is acceptable in field applications. When it comes

Downloaded from ascelibrary.org by Newcastle University on 03/28/19. Copyright ASCE. For personal use only; all rights reserved.

© ASCE

to the six-step prediction, the best CNN model is more accurate than the SVR model, and, because the trained SVR models cannot predict multi-step traffic in Experiment 2, SVR should build 12 independent models (6 for flow and 6 for speed), which requires much more training and testing time.

**Table 3. Comparison of Prediction Performance in Experiment 2**

| Model | | MoE | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Training time (min) | Test time (s) | RMSE | NRMSE (%) | SMAPE 1 (%) | SMAPE 2 (%) | $R^2$ |
| CNN | Flow & speed data | 424.73 | 2.3 | 0.046 | 7.400 | 4.143 | 2.701 | 0.964 |
| | *Flow error* | NA | NA | 0.045 | 9.217 | 5.717 | 3.781 | 0.963 |
| | *Speed error* | NA | NA | 0.047 | 6.451 | 2.570 | 2.066 | 0.893 |
| | Flow data only | 391.69 | 2.1 | 0.047 | 8.659 | 5.553 | 3.274 | 0.974 |
| | Speed data only | 392.17 | 2.0 | 0.044 | 5.682 | 1.965 | 1.557 | 0.944 |
| SVR | Flow & speed data | 596.76 | 99.2 | 0.054 | 8.668 | 6.513 | 3.617 | 0.951 |
| | Flow data only | 296.58 | 51.2 | 0.051 | 10.604 | 9.497 | 4.674 | 0.951 |
| | Speed data only | 300.18 | 48.0 | 0.056 | 7.675 | 3.530 | 2.973 | 0.850 |
| HA | Flow & speed data | NA | 20.0 | 0.072 | 11.521 | 6.461 | 3.870 | 0.913 |
| | Flow data only | NA | 10.1 | 0.072 | 14.805 | 9.606 | 6.079 | 0.905 |
| | Speed data only | NA | 9.9 | 0.072 | 9.752 | 3.316 | 2.568 | 0.756 |

NA: not applicable.

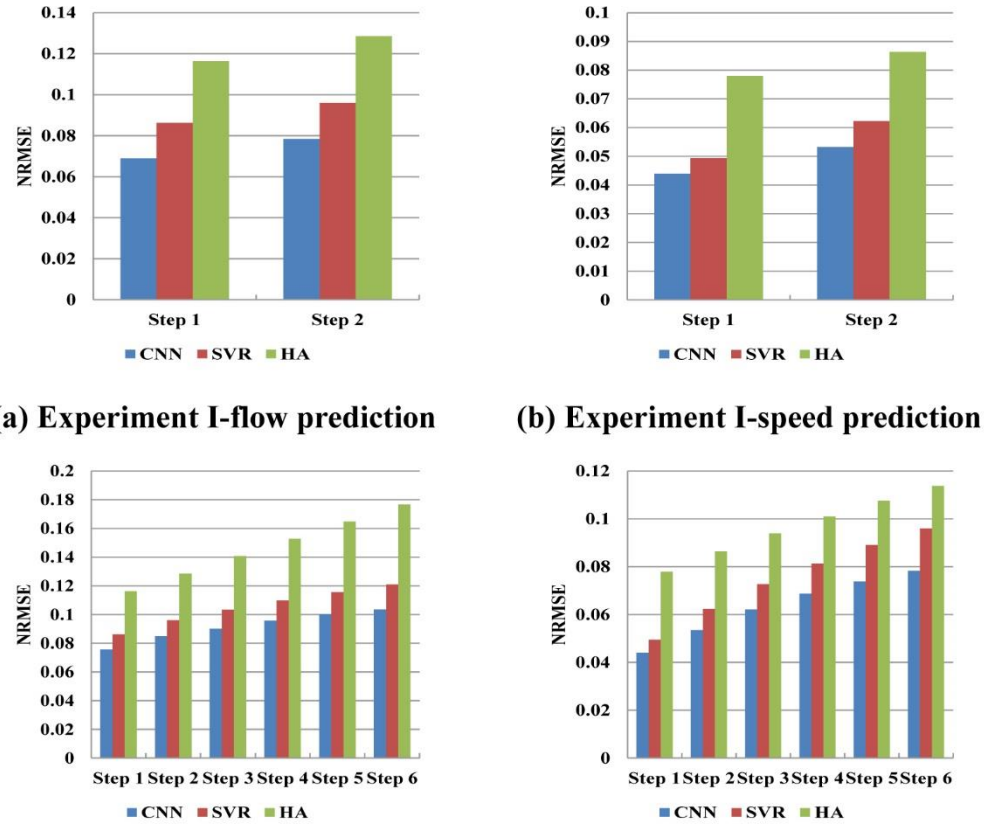
In order to compare the long-term prediction performance of CNN, SVR and HA models, Figure 4 presents the testing NRMSE of the best CNN model, SVR, and HA models with different prediction steps. It can be seen that though the NRMSE of the best CNN model increases with the prediction step, which is as expected, the increasing rate is much lower than the SVR and HA models. That is, in multi-step prediction, the best CNN model outperforms the SVR and HA models. When the prediction step increases (the correlation decreases), the performance of SVR and HA becomes worse while CNN appears more effective for multi-step traffic flow and speed prediction because more temporal and spatial correlations have been taken into account.

## 4 CONCLUSIONS

This paper proposed a CNN-based model to simultaneously predict multi-step traffic speed and flow at different locations considering temporal and spatial correlations. Numerical experiments using multi-detector traffic data from PeMS were conducted to compare the performance of the CNN models with the SVR and HA models. In the experiments, 28-day data collected from 57 detector stations were used. The results show that the best CNN model considering multi-detector traffic speed and flow together outperforms the SVR and HA models which only consider speed or flow while ignoring spatial correlations of multiple traffic detectors. The best CNN model can save much training and predicting time. Meanwhile, the best CNN model has a better performance in multi-step traffic prediction than the benchmark models.

Downloaded from ascelibrary.org by Newcastle University on 03/28/19. Copyright ASCE. For personal use only; all rights reserved.

© ASCE




**(a) Experiment I-flow prediction** **(b) Experiment I-speed prediction**

**(c) Experiment II-flow prediction** **(d) Experiment II-speed prediction**

**Figure 4. Comparison of multi-step prediction errors of different models**

There are still some limitations in this study. The current work only considers detectors installed on one freeway segment. In the future work, we will consider more loop detectors in the freeway network. The traffic density or time occupancy will also be taken into consideration in the CNN models. Besides, more features should also be incorporated so as to improve the prediction performance.

## ACKNOWLEDGEMENTS

This research is financially supported by Zhejiang Provincial Natural Science Foundation of China [LR17E080002], National Natural Science Foundation of China [51508505, 71771198, and 51338008], Fundamental Research Funds for the Central Universities [2017QNA4025], and the Key Research and Development Program of Zhejiang [2018C01007].

## REFERENCES

Castro-Neto, M., Jeong, Y. S., Jeong, M. K. et al. (2009). "Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions." *Expert Systems with Applications*, 36(3), 6164-6173.

Davis, G. A. and Nihan, N. L. (1991). "Nonparametric regression and short-term freeway traffic forecasting." *J. of Transportation Engineering*, 117(2), 178-188.

Hong, W. C. (2011). "Traffic flow forecasting by seasonal SVR with chaotic simulated annealing algorithm." *Neurocomputing*, 74(12), 2096-2107.

© ASCE



Downloaded from ascelibrary.org by Newcastle University on 03/28/19. Copyright ASCE. For personal use only; all rights reserved.

Ke, J., Zheng, H., Yang, H. et al. (2017). "Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach." *Transportation Research Part C*, 85, 591-608.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). "Imagenet classification with deep convolutional neural networks." *Adv. in Neural Information Processing Systems* (pp. 1097-1105).

LeCun, Y., Boser, B., Denker, J. S. et al. (1989). "Backpropagation applied to handwritten zip code recognition." *Neural Computation*, 1(4), 541-551.

Li, L., Chen, X., and Zhang, L. (2014). "Multimodel ensemble for freeway traffic state estimations." *IEEE Transactions on Intelligent Transportation Systems*, 15(3), 1323-1336.

Li, L., He, S., Zhang, J. et al. (2016). "Short-term highway traffic flow prediction based on a hybrid strategy considering temporal-spatial information." *J. of Advanced Transportation*, 50(8), 2029-2040.

Lv, Y., Duan, Y., Kang, W. et al. (2015). "Traffic flow prediction with big data: a deep learning approach." *IEEE Transactions on Intelligent Transportation Systems*, 16(2), 865-873.

Ma, X., Dai, Z., He, Z. et al. (2017). "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction." *Sensors*, 17(4), 818.

Min, W. and Wynter, L. (2011). "Real-time road traffic prediction with spatio-temporal correlations." *Transportation Research Part C*, 19(4), 606-616.

Polson, N. G., & Sokolov, V. O. (2017). "Deep learning for short-term traffic flow prediction." *Transportation Research Part C*, 79, 1-17.

Wu, C. H., Ho, J. M., and Lee, D. T. (2004). "Travel-time prediction with support vector regression." *IEEE Transactions on Intelligent Transportation Systems*, 5(4), 276-281.

Xia, D., Wang, B., Li, H. et al. (2016). "A distributed spatial-temporal weighted model on MapReduce for short-term traffic flow forecasting." *Neurocomputing*, 179, 246-263.

Zhang, J., Zheng, Y., & Qi, D. (2017, February). "Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. " *AAAI*, (pp. 1655-1661).

Zhang, Y. and Haghani, A. (2015). "A gradient boosting method to improve travel time prediction." *Transportation Research Part C*, 58, 308-324.

Downloaded from ascelibrary.org by Newcastle University on 03/28/19. Copyright ASCE. For personal use only; all rights reserved.

© ASCE