Research on Lane Occupancy Rate Forecasting Based on the Capsule Network

RAN TIAN, JIAMING BI, QIANG ZHANG, AND YANXING LIU
Department of Computer Science and Engineering, Northwest Normal University, Lanzhou 730070, China
Corresponding author: Ran Tian (troom@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 71961028 and Grant 71764025, in part by the Scientific Research Projects of Colleges and Universities in Gansu Province under Grant 2019B-038, in part by the Scientific Research Project of the Lanzhou Science and Technology Program under Grant 2018-1-58, and in part by the Scientific Research Project of Northwest Normal University under Grant NWNU-LKQN-18-25.

ABSTRACT This paper proposes a hybrid lane occupancy rate prediction model called 2LayersCapsNet, which combines the improved capsule network and convolutional neural networks (CNNs). The model uses CNNs to mine the spatial-temporal correlation characteristics of the lane occupancy rate and then uses an improved capsule network to mine the interrelationships in traffic data measured by sensors in continuous time intervals to predict the lane occupancy rate. The model can solve the problem of CNNs losing important spatiotemporal information caused by the maximum pooling operation and can obtain better prediction results. To verify the efficiency of the 2LayersCapsNet model, the model is compared with the capsule network model (CapsNet), convolutional neural networks model (CNNs), recurrent neural networks model (RNNs), long short-term memory model (LSTM) and stacked autoencoders model (SAEs) with similar network structures and parameter settings on the PEMS-SF data set. The experimental results indicate that the 2LayersCapsNet model can obtain a prediction model faster than the CapsNet model and that 2LayersCapsNet performs better than the CNNs, RNNs, LSTM and SAEs with respect to three evaluation metrics, namely, MAPE, MAE and RMSE, on four prediction tasks.

INDEX TERMS Lane occupancy rate, forecasting, capsule network, convolutional neural network.

I. INTRODUCTION

Due to the growing population and substantially increased automobile usage, traffic jams and accidents caused by illegal occupation of lanes have become a major problem around the world [1]. Intelligent transportation systems (ITS) have been developed to solve this problem [2]. Lane occupancy rate forecasting, a key component of ITS [3], can provide real-time, accurate and fast traffic guidance management and traffic flow control and is also an effective way to alleviate urban traffic congestion, reduce vehicle exhaust emissions, and prevent and address traffic accidents. Therefore, lane occupancy rate forecasting has important research significance [4]. Lane occupancy rate forecasting is different from traffic flow prediction: attention is focused on traffic congestion and traffic accidents caused by lane occupancy that result in personal injury [5].

The urban traffic system is open and complex, which makes lane occupancy rates subject to external factors, including weather conditions, traffic accidents, and traffic control. An analysis of a large quantity of traffic detector data has shown that the lane occupancy rate has periodicity and similarity on long time scales but is time-varying, chaotic and correlated on short time scales [6]. Based on these characteristics, experts and scholars mainly use nonlinear mapping models [7] to predict the short-term lane occupancy rate. Short-term lane occupancy rate forecasting is similar to traffic flow prediction, which has achieved rich research results in recent years. The commonly used methods include mathematical models such as historical average time series models [8] and Kalman state space filtering models [8]; and prediction methods based on nonlinear theories such as the ARIMA model [10], fuzzy logic model [11], K-nearest neighbor model [12], Bayesian network model [13], back propagation neural network model [14], radial basis function neural networks [15], wavelet theory [16], support vector machines [17], and Gaussian process methods [18].
In these forecasting methods, neural networks have become a research hotspot because of their large parallel structure and distributed storage characteristics. Neural network models, such as DBM, RBF, RNN and CNN, are used to solve forecasting problems. Because these in-depth learning models are characterized by powerful function approximation ability and pattern classification ability and good self-organization, self-adaptability and fault tolerance, they have achieved many good results. For example, Yang et al. [19] proposed the stacked autoencoder Levenberg-Marquardt model, which is a deep neural network approach that aims to improve the forecasting accuracy. Huang et al. [20] proposed a two-part deep architecture, first using an underlying deep belief network (DBN) for unsupervised feature learning and then using the multitasking regression layer on the DBN for supervisory prediction. Lv et al. [21] proposed a deep hierarchical structure model based on large flow data that uses an automatic encoder as a building block to represent traffic flow characteristics for prediction. Du et al. [22] proposed a multilayer integrated deep learning architecture that obtains long-term trend features by means of recursive neural networks (RNNs) and local trend features by means of convolutional neural networks (CNNs). Wu and Tan [23] used a one-dimensional convolutional network to capture the spatial characteristics of traffic flow, used two long short-term memory methods to mine the trend features by means of recursive neural networks (RNNs) and local features by means of convolutional neural networks (CNNs). Wang et al. [24] proposed a traffic flow velocity prediction model with single hidden layer CNNs combined with error feedback. Ma et al. [25] proposed a new method of transforming traffic flow into an image. In this method, the traffic speed data of each section in each time step are expressed in three dimensions, and the space-time features of images are captured by CNNs. This method has been proven to be superior to other advanced methods. These models have achieved good results in traffic flow forecasting using a deep learning algorithm with the ability to approximate arbitrary nonlinear functions and self-learning. Some other models combine prediction models with other factors such as weather [26] and use deep neural networks to produce good results.

However, in practical applications, the existing deep learning models have two problems. On the one hand, the time characteristics of the lane occupancy rate are difficult to mine. Under special circumstances, weather conditions, traffic accidents, traffic control and other factors will have different impacts on the occupancy rate of different lanes, which makes the lane occupancy rate abnormal. However, most existing deep learning models are based on the principle of empirical risk minimization. Their structure selection lacks theoretical guidance, and it is easy to fall into local extrema. For example, in RBF neural networks, the selection of a central vector and the number of hidden layer neurons has a considerable impact on the learning and generalization ability. However, the correlation between the spatial-temporal characteristics of the lane occupancy rate are difficult to extract with existing deep learning algorithm models. In terms of time characteristics, there is a relationship between the occupancy rate of a lane and that of its adjacent lanes. Existing deep learning models typically simulate the time dimension by increasing the depth of the model. For example, a deep belief neural network (DBN) can only train on one-dimensional data, not two-dimensional space-time data, and unreasonable hyperparameter settings will negatively affect the performance of the network. In terms of spatial characteristics, the lane occupancy information states of the sections and their upstream and downstream will influence each other. However, the existing deep learning models can only identify the existence of features because the input is scalar, and the spatial features between inputs are difficult to analyze. For example, in CNNs, the maximum pooling layer loses valuable information and ignores the correlation between local and global features.

CapsNet is a neural network model proposed by Sabour et al. [27] that uses the concept of a capsule to overcome the characterization limitations of CNNs and RNNs. CapsNet uses a vector output capsule instead of the scalar output feature detector of a CNN, uses dynamic routing instead of maximum pooling, and achieves good results on MNIST data sets. Kim et al. [28] applied CapsNet to traffic flow forecasting and achieved better results than those obtained using CNNs. However, because of its dynamic routing mechanism, CapsNet is 30 times slower than CNNs. Moreover, a prediction model can only output one predicted value of a time unit. If we need to predict the lane occupancy rate for a period of time (such as \( n \) time units), \( n \) models must be trained. Therefore, from the perspective of improving the speed of lane occupancy prediction, this paper proposes a prediction model via which multiple prediction periods can be obtained simultaneously based on a capsule network.

The main contributions of the paper are the following:

1) An improved CapsNet algorithm is proposed.
2) A lane occupancy prediction model called 2LayersCapsNet is proposed. The experimental results show that 2LayersCapsNet can obtain a prediction model with 90% accuracy within 4 minutes, while the traditional single-layer capsule network model (CapsNet) takes approximately 9 minutes: 2LayersCapsNet’s convergence speed is 120% faster than that of CapsNet.
3) The experimental results of four prediction tasks show that 2LayersCapsNet is essentially equivalent to CapsNet in terms of three evaluation metrics (MAPE, MAE, RMSE) and is superior to CNN and LSTM models with similar network structures and parameter settings.

II. LANE OCCUPANCY RATE FORECASTING MODEL

In this section, we propose the 2LayersCapsNet model for lane occupancy rate forecasting. We first define the lane occupancy rate input matrix and output vector based on spatial-temporal characteristics (Section A). Then, we review the
traditional capsule network and change its routing component to make it more suitable for regression tasks. (Section B). Finally, the detailed architecture of the 2LayersCapsNet prediction model is introduced (Section C).

A. INPUT MATRIX AND OUTPUT VECTOR OF THE LANE OCCUPANCY RATE BASED ON SPATIOTEMPORAL CHARACTERISTICS

Because of the spatial-temporal correlation of the lane occupancy rate, we must consider the traffic information status of the forecasting segment and its upstream and downstream when constructing the forecasting vector. The lane occupancy rate follows some change rules over time: the lane occupancy rate is affected by upstream and downstream traffic conditions in space and shows a certain correlation: the downstream lane occupancy rate can be estimated by the upstream lane occupancy rate. In this paper, the original lane occupancy rate is processed into a lane occupancy rate matrix with temporal and spatial information based on the spatial-temporal characteristics. For \( S \) monitoring points in \( T \) time periods, the input matrix \( X \) can be expressed as follows:

\[
X = \begin{bmatrix}
  x_{1,1} & x_{1,2} & \cdots & x_{1,T} \\
  x_{2,1} & x_{2,2} & \cdots & x_{2,T} \\
  \cdots & \cdots & \cdots & \cdots \\
  x_{S,1} & x_{S,2} & \cdots & x_{S,T}
\end{bmatrix}
\]

(1)

Then, the lane occupancy rate in \( n \) periods after the predicted monitoring point \( s \) can be expressed as the output vector \( y \):

\[
y = [x_{s,T+1}, x_{s,T+2}, \ldots, x_{s,T+n}]
\]

(2)

Assuming that the lane occupancy rate of monitoring point \( s \) at time \( t \) is represented by \( x_{s,t} \), the data monitoring point \( s \) is assumed to be related only to the flow of its 2 upstream data monitoring points in space. Suppose we need to predict the lane occupancy rate at point \( s \) for one hour, which is related to the lane occupancy rate of the previous four hours. Assuming that the lane occupancy rate is measured every 10 minutes, the input matrix of the prediction can be expressed as:

\[
X = \begin{bmatrix}
  x_{s,t} & x_{s,t+1} & \cdots & x_{s,t+23} \\
  x_{s+1,t} & x_{s+1,t+1} & \cdots & x_{s+1,t+23} \\
  x_{s+2,t} & x_{s+2,t+1} & \cdots & x_{s+2,t+23}
\end{bmatrix}
\]

(3)

The predicted vector can be expressed as:

\[
y = [x_{s,t+24}, x_{s,t+25}, \ldots, x_{s,t+29}]
\]

(4)

B. CAPSULE NETWORK MODEL

The original goal of CapsNet was to solve the target recognition task, and CapsNet demonstrates excellent performance on the MNIST data set. A capsule is the basic element of the capsule network, similar to the neuron of the traditional neural network. Unlike traditional neurons, capsules are composed of a group of neurons and are a feature vector that is used as input and output of CapsNet. Traditional neural networks use affine transformation and nonlinear activation functions (such as ReLU, sigmoid, and tanh) to transform low-level neurons into high-level neurons. CapsNet first uses affine transformation to transform low-level neurons into high-level neurons and then updates the weight by applying a dynamic routing algorithm between two continuous capsules, thereby increasing or reducing the connection strength between multiple input and output vectors, which is more effective than the basic routing strategy in CNNs. In contrast to the maximum pooling layer commonly used in CNNs, CapsNet does not drop information that may be related to output, such as the correlation between the temporal and spatial characteristics of the lane occupancy rate.

Assume that the low-level input capsule is defined as \( u = (u_1, u_2, \ldots, u_n) \), \( u_i \in \mathbb{R}^d \) is the \( i \)th capsule of input. The high-level input capsule is defined as \( v = (v_1, v_2, \ldots, v_m) \), \( v_j \in \mathbb{R}^p \) is the \( j \)th capsule of output. The mechanism of dynamic routing can be divided into two steps:

**Step 1: Affine transformation.** Traditional CapsNet obtains the intermediate eigenvector \( \hat{u}_{ij} \) by multiplying the input vector \( u_i \) and the weight matrix \( W_{ij} \).

\[
\hat{u}_{ij} = W_{ij} \cdot u_i
\]

(5)

\( i = [1,n], j = [1,m] \). The weight matrix \( W_{ij} \) is a matrix of \( n \times m \), as shown in Fig. 1 (a).

To avoid the overfitting phenomenon caused by a large number of matrix parameters and improve the calculation speed of the model, this article uses a matrix weight sharing mechanism [29] to reduce weights and accelerate the calculations. In the weight sharing mechanism shown in Fig. 1 (b), the shared weight matrix of the input capsule and the output capsule connected to the lower layer is \( W_j \), and the intermediate feature vector \( \hat{u}_{ij} \) is:

\[
\hat{u}_{ij} = W_j \cdot u_i
\]

(6)

**Step 2: Dynamic routing.** Traditional CapsNet uses dynamic routing to calculate the output vector. The core of CapsNet is to update parameters according to its own
Algorithm 1 Dynamic Routing

Define Dynamic_Routing \((u_{ji}, r, l)\):

\[
\hat{u}_j = k\_means(\hat{u}_{ji})
\]

\[
b_{ji} = (\hat{u}_{ji}, \hat{u}_j)
\]

\[
c_{ji} = \text{Leaky}_\text{Relu}(b_{ji}, \text{alpha} = 0.2)
\]

For \( i \) in \( r \) iterations do

for all capsule \( i \) in layer \( l \) and capsule \( j \) in layer \( l + 1 \):

\[
c_{ji} = \text{Leaky}_\text{Relu}(b_{ji}, \text{alpha} = 0.2)
\]

for capsule \( j \) in layer \( l + 1 \):

\[
s_j = \sum_{i=1}^{m} \hat{u}_{ji} c_{ji}, v_j = \text{squash}(s_j)
\]

Return \( v_j \)

The initial characteristics by iteration of dynamic routing to obtain the output vector that best reflects the characteristics of the intermediate eigenvectors. However, the traditional routing parameters are usually initialized with uniform distribution, ignoring the differences among low-level capsules. By contrast, the routing parameters can be initialized via k-means clustering to improve the routing process. In the squashing function, we refer to ReLU. We set the weights of short vectors to 0, so these vectors are not involved in the next dynamic routing calculation process, thereby reducing the training time. The calculation process of dynamic routing is shown in Algorithm 1.

First, we use k-means clustering to calculate the initial \( c_{ji} \) of the low-level capsules. Before the iteration begins, the following steps are performed:

(1) Calculate the cluster center \( \hat{u}_j \) of \( \hat{u}_{ji} \).

\[
\hat{u}_j = k\_means(\hat{u}_{ji})
\]

(2) Initialize \( b_{ji} \) as the distance from each \( \hat{u}_{ji} \) to the cluster center \( \hat{u}_j \).

\[
b_{ji} = (\hat{u}_{ji}, \hat{u}_j)
\]

The vector distance \((\hat{u}_{ji}, \hat{u}_j)\) is calculated as the cosine similarity.

(3) Calculate the weight \( c_{ji} \) of each \( \hat{u}_{ji} \).

\[
c_{ji} = \text{Leaky}_\text{Relu}(b_{ji}, \text{alpha} = 0.2)
\]

\( c_{ji} \) indicates the degree of association between low-level capsule \( u_{ji} \) and cluster center \( \hat{u}_j \).

\[
s_j = \sum_{i=1}^{m} \hat{u}_{ji} c_{ji}
\]

Before the first iteration, the traditional routing algorithm in CapsNet initializes \( b_{ji} \) to 0. Because \( c_{ji} = \text{Softmax}(b_{ji}) \), the initial \( c_{ji} \) is the same, and the route in the first iteration does not play a substantial role, as shown in Fig. 2(a). Therefore, we use k-means clustering to obtain the cluster center \( \hat{u}_j \) and initialize \( b_{ji} \) based on the distance between \( \hat{u}_{ji} \) and \( \hat{u}_j \) to obtain the value of \( c_{ji} \) so that the first iteration plays an important role, as shown in Fig. 2(b). At the same time, we change the softmax function to the Leaky Relu function because the goal is to predict rather than classify. Moreover, the LeakyRelu function results in faster computing speeds and better results.

Second, we use a nonlinear squashing function to ensure that the short vector can be compressed to 0, and the long vector can be completely saved to obtain a high level of capsules \( v_j \)

\[
v_j = \text{squash}(s_j) = \begin{cases} 0, & ||S|| \leq \alpha \\ \frac{||S||^2}{1+||S||^2} \cdot S, & ||S|| > \alpha \end{cases}
\]

\( \alpha \) is defined as 1/4 of the maximum length of the output vector. If the vector is short, there are insufficient features in the vector, and there is no need to reserve them. According to the obtained \( v_j \), the similarity \((\hat{u}_{ji}, v_j)\) of \( \hat{u}_{ji} \) and \( v_j \) is calculated, and the next iteration parameter \( c_{ji} \) is obtained by Leaky_ReLU.

\[
b_{ji} = b_{ji} + (\hat{u}_{ji}, v_j)
\]

\[
c_{ji} = \text{Leaky}_\text{Relu}(b_{ji}, \text{alpha} = 0.2)
\]

Then, the next iteration is performed based on \( c_{ji} \). After all iterations are completed, output \( v_j \) is the calculation result of the dynamic route.

In summary, compared with traditional CapsNet, clustering is introduced in the dynamic route, and Leaky_ReLU is used instead of softmax. Experiments show that such changes improve the performance of CapsNet on regression tasks.

C. PREDICTION MODEL

The capsule network regards a feature as an active vector that can be used in many time series prediction tasks. In general, the input of many prediction tasks is a time series representing historical values. We can assign each historical value in the sequence to each region and then transform the region into the distribution representation of the region. The distribution of the region can be regarded as a vector such that the sequence of such regions can be regarded as a group of capsules. According to the characteristics of the input and output vectors of the capsule network, we can use the capsule network to process the time series prediction task and obtain
the output of multiple prediction values in one calculation. Therefore, two prediction models are designed in this paper. The first prediction model, CapsNet, uses a basic model similar to text recognition (Fig. 3). The second prediction model, 2LayersCapsNet, takes into account the spatial-temporal correlation of the lane occupancy rate (Fig. 4) and uses a convolutional layer to acquire spatial-temporal characteristics to enhance the capsule layer.

1) CONSTRUCTION OF THE CONVOLUTIONAL LAYER (Conv_Layer)

The purpose of constructing the convolutional layer Conv_Layer is to obtain a more abstract feature. The convolutional layer $H_1$ is calculated by convolution and contains many convolution surfaces. The $a$-th convolution surface of the convolutional layer $H_1$ is denoted by $h_{1,a}$, $w^1$ is the convolution kernel matrix, and the activation function is ReLU.

$$h_{1,a} = \text{ReLU} \left( \sum_{s=1}^{\text{nums}} \sum_{t=1}^{\text{time}} x_{s,t} \cdot w^1_{s,t} + b_{1,a} \right)$$

(14)

$x_{s,t}$ is the value of the convolution kernel in node $(s,t)$ and $\cdot$ represents point multiplication. The input matrix is $[\text{station_nums}, \text{train_time_interval}]$, $\text{station_nums}$ is the prediction site and $\text{station_nums} - 1$ is the site before the prediction site. $\text{train_time_interval}$ is the predicted time period of the station before the predicted time node. In the CapsNet model, the convolution kernel size is $[3,3]$, the stride is 1, and the number of convolution kernels is 150. In the 2LayersCapsNet model, the convolution kernel size of the first layer is $[3, \text{train_time_interval}]$, the convolution kernel size of the second layer is $[\text{station_nums}, 3]$, all layers have a stride of 1, and the number of convolution kernels is $\text{pred_time_interval} \times 32(\text{pred_time_interval} \text{ is the number of time periods to be predicted})$.

2) CONSTRUCTION OF PRIMARYCAPSLAYER

The purpose of PrimaryCaps is to reconstruct the input data of the convolutional layer into vector data with a characteristic direction. The construction process is divided into two steps: construct the convolutional capsule layer and calculate the dynamic routing. The process is shown in Fig. 5.

The convolution process of the first step of PrimaryCapsLayer is similar to that of Conv_Layer. The activation function of the CapsNet model is ReLU, the size of the convolution kernel is $[3,3]$, the stride is 2, and the convolutional output is reconstructed into many $\text{pred_time_interval}$ dimension vectors with 32 channels (each vector contains $\text{pred_time_interval}$ features). The second step is to calculate the process of dynamic routing. The detailed calculation process is shown in Algorithm 1. Finally, the output capsule that best reflects the input characteristics is output to the DigitCapsLayer. The CapsNet and 2LayersCapsNet models have the same activation function and output shape, but the convolution cores of 2LayersCapsNet are $[2, \text{train_time_interval}]$ and $[\text{station_nums}, 2]$ in 2 layers (Fig. 4).

Finally, the output vectors are obtained by the weighting calculation after the dynamic routing calculation is completed and output to the DigitCapsLayer. The dynamic routing output of the first layer of 2LayersCapsNet is $v^1 = (v^1_1, v^1_2, \ldots, v^1_m)$, the layer 2 dynamic routing output is $v^2 = (v^2_1, v^2_2, \ldots, v^2_n)$, and the output of PrimaryCapsLayer is obtained as the weighted summation of the outputs of the two layers (Formula 13).

$$v = \sum W_i \cdot v_i + b_i$$

(15)

3) CONSTRUCTION OF DIGITCAPSLAYER

The purpose of DigitCapsLayer is to reconstruct the vector data with feature direction into prediction data through three fully connected layers, as shown in Fig. 4:

First, the data output from PrimaryCapsLayer is flattened. Then, the data pass through 3 fully connected layers. The outputs of the three fully connected layers are 256,
512 and \(\text{pred}_\text{time}_\text{interval}\), and the final output layer gives \(\text{pred}_\text{time}_\text{interval}\) predicted values.

4) LOSS FUNCTION
To minimize the error loss \(\text{Loss}(y, \hat{y})\) between the true value \(y\) and the predicted value \(\hat{y}\), the following formula is established:

\[
\theta = \arg \min \theta \text{Loss}(y, \hat{y}) = \arg \min \theta \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|
\]  \hfill (16)

5) CALCULATION OF PARAMETERS
In the training process of the network, parameters \(W^{L_1}\) must be determined in the convolutional layer \((L_1:\text{Conv}_\text{Layer})\), two parameters \(c^{L_2}\) and \(W^{L_2}\) must be determined in PrimaryCapsLayer \((L_2)\), and parameters \(W^{L_3}\) and \(b^{L_3}\) must be determined in DigitCapsLayer \((L_3)\). \(c^{L_2}\) can be calculated by dynamic routing, and the other parameters can be calculated by the adaptive moment estimation (Adam) gradient descent algorithm. Adam uses the first moment estimate and the second moment estimate of the gradient to dynamically adjust the learning rate of each parameter. After bias correction, the learning rate of each iteration is within a certain range, which makes the parameters more stable. This process is conducive to fully extracting the lane occupancy rate characteristics [30].

III. EXPERIMENTS
A. DATA DESCRIPTION
The experimental data in this paper are from PEMS-SF (http://archive.ics.uci.edu/ml/datasets/PEMS-SF) and are obtained from the PEMS website of the California Department of Transportation. The data represent the occupancy rate, between 0 and 1, of different car lanes of San Francisco Bay area freeways. The data cover 440 days from January 1, 2008 to March 30, 2009, with samples taken every 10 minutes. The data set considers each day as a 963-dimensional single time series (i.e., the number of sensors that work continuously throughout the study period). The training data cover 267 days, and the test data cover 173 days. As an example, we consider the first sensor and present its lane occupancy rate data in Fig. 7.

As shown in Fig. 7, the lane occupancy rate data are concentrated in \((0,0.1)\), and the maximum value does not exceed 0.5. Moreover, Fig. 8 shows that the lane occupancy rate changes of adjacent sensors have a causal relationship, so the influence of peripheral sensors on lane occupancy rate prediction of a sensor cannot be ignored.

B. EXPERIMENTAL PLATFORM AND PREDICTION TASK
The experimental platform of this paper is an Intel i7-6700K, 16 GB. The project is based on TensorFlow 1.13 and python 3.6 and is programmed in PyCharm. Time series prediction tasks use historical data to predict future data. The four prediction tasks are defined as follows:

- **TASK 1**: 30-min prediction with 240-min traffic history on 5 road segments;
- **TASK 2**: 60-min prediction with 240-min traffic history on 5 road segments;
- **TASK 3**: 90-min prediction with 240-min traffic history on 5 road segments;
- **TASK 4**: 120-min prediction with 240-min traffic history on 5 road segments;

C. EVALUATION METRICS
The mean absolute percentage error (MAPE), mean absolute error (MAE) and root mean square error (RMSE) are used to evaluate the prediction results.

\[
\text{MAPE} = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad (17)
\]

\[
\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \quad (18)
\]

\[
\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \quad (19)
\]
FIGURE 9. Comparison of 6 predicted values of 60 traffic monitoring points with real values: CapsNet and 2LayersCapsNet.

FIGURE 10. Comparison of prediction results of models on Task 2.

$y_i$ denotes the true value, $\hat{y}_i$ denotes the predicted value, and $\bar{y}_i$ denotes the average value of $y_i$. For illustration purposes, the accuracy of prediction is defined as:

$$\text{accuracy} = 100 - \frac{100}{\text{MAPE}}$$  \hspace{1cm} (20)

D. EXPERIMENTAL RESULTS AND ANALYSIS

1) PREDICTION RESULTS OF THE MODELS

From the intuitive prediction results, model CapsNet and model 2LayersCapsNet take 60 monitoring points from the PEMS-SF data set and predict the results on Task 2, as shown in Fig. 9. ($60 \times 6 = 360$ predicted values).

To obtain a clearer view, a random point is selected from the test set to illustrate the prediction results of different models, as shown in Fig. 10.

Fig. 10 shows that the predicted values of 2LayersCapsNet are more accurate than those of CapsNet in reflecting the changing trend of the lane occupancy rate on Task 2. The change in RMSE on the training set and test set is shown in Fig. 11.

Fig. 11 indicates that the RMSEs of the training set and test set are basically consistent, which shows that 2LayersCapsNet does not overfit the data.

2) COMPARISON OF THE EXPERIMENTAL RESULTS OF THE IMPROVED CAPSULE MODEL

The parameters of the CapsNet model are shown in Table 1.

With the same model as that Fig. 2, we compare the traditional method in the CapsNet model with the improved method with shared weight, k-means and a new squashing function in terms of the MAPE of lane occupancy prediction of Task 2. The comparison result is shown in Fig. 13.
3) INFLUENCE OF PARAMETERS ON THE MODEL RESULTS

The change curves of MAPE for the CapsNet model and 2LayersCapsNet model during training are shown in Fig. 12. Fig. 12 shows that the first MAPE value < 10 of 9.58 for 2LayersCapsNet is achieved when Globalsteps = 135 and time = 212.89 s. However, the first MAPE value < 10 of 9.97 for CapsNet is achieved when Globalsteps = 531 and time = 840.23 s. Therefore, the convergence rate of 2LayersCapsNet is faster than that of CapsNet.

The effect of the dynamic routing parameters on 2LayersCapsNet is shown in Fig. 14 (the smoothing function is Savitzky-Golay, points = 200).

The number of iterations for dynamic routing is varied in iter_routing = 1, 3, 5. The convergence speed of RMSE is the fastest when iter_routing = 5; in other words, the greater the number of iterations of dynamic routing is, the greater the number of output vectors, the greater the number of features used to reflect the input data, and the greater the quantity of spatiotemporal information and internal connections that can be found in the traffic data. However, from the perspective of the RMSE convergence rate, the speed is similar when iter_routing = 1 and iter_routing = 3, but when iter_routing = 5, more computing time is required for convergence. In the following experiments, we set iter_routing = 3.

4) RESULTS AND COMPARISON

To verify the validity of the proposed model, five deep learning-based models are chosen for comparison. CapsNet performs regression using the traditional CapsNet network structure shown in Fig. 1. CNNs [22] is a neural network consisting of three convolutional layers, three pooling layers and a fully connected layer. RNN takes into account the context of the data during training and can learn these characteristics by expanding time series, sharing parameters and hiding the state capture patterns at each time step. LSTM NN is an extension of RNNs and has become popular since the architecture can address long-term memories and avoid vanishing gradient issues. The RNN [31] and LSTM [32] models are optimized to contain three hidden layers with 1000 hidden units in each layer. SAEs is a neural network consisting of multiple layers of autoencoders, where the model inputs are encoded into dense or sparse representations before being fed into the next layer, and the SAEs model [18] is tuned to form three autoencoder layers with 3000, 2500 and 2000 hidden units in the three layers. The results for the 4 prediction tasks are shown in Table 2.

Table 2 shows that 2LayersCapsNet is superior to CNNs and LSTM. Fig. 15 shows the prediction error rate (PER) for 6 time intervals at 20 traffic monitoring points (6 × 20 = 120 predicted values).

\[
PER = 100\% \times \frac{|\hat{y} - y|}{y} \tag{21}
\]
**TABLE 2. Comparison of each model on 4 tasks on the test data.**

| Model          | MAPE(%) | RMSE  | CNNs | RNNs     | LSTM    | SAEs     |
|----------------|---------|-------|------|----------|---------|----------|
| TASK1          |         |       |      |          |         |          |
| 2LayersCapsNet | 3.68    | 3.88  | 3.39 | 3.51     | 3.72    | 3.55     |
| CapsNet        | 4.24    | 4.07  | 4.21 | 4.19     | 4.22    | 4.19     |
| CNNs           | 4.37    | 5.04  | 4.46 | 5.03     | 4.52    | 5.09     |
| RNNs           | 4.14    | 4.12  | 4.09 | 4.07     | 4.10    | 4.10     |
| LSTM           | 4.04    | 4.17  | 4.02 | 4.15     | 4.03    | 4.15     |
| SAEs           | 4.21    | 4.24  | 4.15 | 4.20     | 4.16    | 4.21     |
| TASK2          |         |       |      |          |         |          |
| 2LayersCapsNet | 3.68    | 3.88  | 3.39 | 3.51     | 3.72    | 3.55     |
| CapsNet        | 4.24    | 4.07  | 4.21 | 4.19     | 4.22    | 4.19     |
| CNNs           | 4.37    | 5.04  | 4.46 | 5.03     | 4.52    | 5.09     |
| RNNs           | 4.14    | 4.12  | 4.09 | 4.07     | 4.10    | 4.10     |
| LSTM           | 4.04    | 4.17  | 4.02 | 4.15     | 4.03    | 4.15     |
| SAEs           | 4.21    | 4.24  | 4.15 | 4.20     | 4.16    | 4.21     |
| TASK3          |         |       |      |          |         |          |
| 2LayersCapsNet | 3.68    | 3.88  | 3.39 | 3.51     | 3.72    | 3.55     |
| CapsNet        | 4.24    | 4.07  | 4.21 | 4.19     | 4.22    | 4.19     |
| CNNs           | 4.37    | 5.04  | 4.46 | 5.03     | 4.52    | 5.09     |
| RNNs           | 4.14    | 4.12  | 4.09 | 4.07     | 4.10    | 4.10     |
| LSTM           | 4.04    | 4.17  | 4.02 | 4.15     | 4.03    | 4.15     |
| SAEs           | 4.21    | 4.24  | 4.15 | 4.20     | 4.16    | 4.21     |
| TASK4          |         |       |      |          |         |          |
| 2LayersCapsNet | 3.68    | 3.88  | 3.39 | 3.51     | 3.72    | 3.55     |
| CapsNet        | 4.24    | 4.07  | 4.21 | 4.19     | 4.22    | 4.19     |
| CNNs           | 4.37    | 5.04  | 4.46 | 5.03     | 4.52    | 5.09     |
| RNNs           | 4.14    | 4.12  | 4.09 | 4.07     | 4.10    | 4.10     |
| LSTM           | 4.04    | 4.17  | 4.02 | 4.15     | 4.03    | 4.15     |
| SAEs           | 4.21    | 4.24  | 4.15 | 4.20     | 4.16    | 4.21     |

**FIGURE 15. Comparison of the prediction error rate between CNNs and 2LayersCapsNet with iter_routing = 3 on Task 1.**

Fig. 15 shows that the PER of 2LayersCapsNet is usually lower than that of CNNs. In other words, 2LayersCapsNet can obtain more accurate prediction values than those produced by CNNs.

In summary, three conclusions can be drawn from Table 2, Fig. 12, Fig. 13 and Fig. 15.  
1) Fig. 12 shows that the MAPE of 2LayersCapsNet converges faster than that of CapsNet, and the other two evaluation metrics (MAE and RMSE) have similar convergence curves; that is, 2LayersCapsNet trains models faster than does CapsNet.

2) Fig. 13 shows that when iter_routing = 1 and iter_routing = 3, the training time of the model is similar. When iter_routing = 5, the training time increases substantially, but the final RMSE convergence interval is not significantly improved. Therefore, iter_routing = 1 or 3 can be used when training 2LayersCapsNet.

3) Table 2 shows that 2LayersCapsNet is superior to CNNs, RNNs, LSTM and SAEs with respect to the three evaluation metrics for the four prediction tasks. Furthermore, Fig. 15 shows that 2LayersCapsNet has better prediction accuracy than CNNs.

**IV. CONCLUSION**

In this paper, we propose a lane occupancy rate prediction model, called 2LayersCapsNet, based on a capsule network and CNN. The model uses CNNs to mine the spatiotemporal correlation characteristics of the lane occupancy rate and then uses the capsule network to mine the interrelationships in traffic data measured by sensors in continuous time intervals to predict the lane occupancy rate. The model can solve the problem of CNNs losing important spatiotemporal information caused by the maximum pooling operation and can obtain better prediction results. Experiments on PEMS-SF data sets show that the convergence rate of 2LayersCapsNet is faster than that of CapsNet. Finally, we compare the model with the CNN model and LSTM model. The experimental results show that 2LayersCapsNet substantially improves the prediction accuracy, which demonstrates the feasibility of the model for lane occupancy rate prediction. Determining how to combine CapsNet with 2LayersCapsNet to improve the generalization ability and application scope of the model is our next goal.

**REFERENCES**

[1] F. Chen, M. Song, and X. Ma, “Investigation on the injury severity of drivers in rear-end collisions between cars using a random parameters bivariate ordered probit model,” *Int. J. Environ. Res. Public Health*, vol. 16, no. 14, p. 2632, Jul. 2019.

[2] G. He and S. Ma, “A study on the short-term prediction of traffic volume based on wavelet analysis,” in *Proc. IEEE 5th Int. Conf. Intell. Transp. Syst.*, Sep. 2002, pp. 731–735.

[3] H. Tan, Y. Wu, B. Shen, P. J. Jin, and B. Ran, “Short-term traffic prediction based on dynamic tensor completion,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2123–2133, Aug. 2016.

[4] K. Guan, B. Ai, A. Fricke, D. He, Z. Zhong, D. W. Matolak, and T. Kurner, “Excess propagation loss modeling of semiclosed obstacles for intelligent transportation system,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2171–2181, Aug. 2016.

[5] F. Chen and S. Chen, “Injury severities of truck drivers in single- and multi-vehicle accidents on rural highways,” *Accident Anal. Prevention*, vol. 43, no. 5, pp. 1677–1688, Sep. 2011.

[6] K. Y. Chan, T. S. Dillon, J. Singh, and E. Chang, “Neural-network-based models for short-term traffic flow forecasting using a hybrid exponen
tial smoothing and Levenberg–Marquardt algorithm,” *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 644–654, Jun. 2012.

[7] F. Chen, S. Chen, and X. Ma, “Analysis of hourly crash likelihood using unbalanced panel data mixed logit model and real-time driving environmental big data,” *J. Saf. Res.*, vol. 65, pp. 153–159, Jun. 2018.

[8] C. Dong, C. Shao, S. H. Richards, and L. D. Han, “Flow rate and time mean speed predictions for the urban freeway network using state space models,” *Transp. Res. C, Emerg. Technol.*, vol. 43, pp. 20–32, Jun. 2014.

[9] B. Ait-Er-Fquih and I. Hoteit, “Fast Kalman-like filtering for large-dimensional linear and Gaussian state-space models,” *IEEE Trans. Signal Process.*, vol. 63, no. 21, pp. 5853–5867, Nov. 2015.

[10] S. V. Kumar and L. Vanajakshi, “Short-term traffic flow prediction using seasonal ARIMA model with limited input data,” *Eur. Transp. Res. Rev.*, vol. 7, no. 3, p. 21, Jun. 2015.

[11] L. Li, W.-H. Lin, and H. Liu, “Type-2 fuzzy logic approach for short-
term traffic forecasting,” *IEEE Proc.-Intell. Transp. Syst.*, vol. 53, no. 1, pp. 33–40, 2006.

[12] H. A. Fayed and A. F. Atiya, “A novel template reduction approach for the K-nearest neighbor method,” *IEEE Trans. Neural Netw.*, vol. 20, no. 5, pp. 890–896, May 2009.

[13] M. Gui, A. Palwak, and S. Das, “Bayesian network model with Monte Carlo simulations for analysis of animal-related outages in overhead distribution systems,” *IEEE Trans. Power Syst.*, vol. 26, no. 3, pp. 1618–1624, Aug. 2011.
[14] C. Li, Y. Xie, H. Zhang, and X.-L. Yan, “Dynamic division about traffic control sub-area based on back propagation neural network,” in Proc. 2nd Int. Conf. Intell. Hum.-Mach. Syst. Cybern., Aug. 2010, pp. 22–25.

[15] D. Chen, “Research on traffic flow prediction in the big data environment based on the improved RBF neural network,” IEEE Trans Ind. Informat., vol. 13, no. 4, pp. 2000–2008, Aug. 2017.

[16] H. Zhang, X. Wang, J. Cao, M. Tang, and Y. Guo, “A multivariate short-term traffic flow forecasting method based on wavelet analysis and seasonal time series,” Int. J. Speech Technol., vol. 48, no. 10, pp. 3827–3838, Apr. 2018.

[17] Y. Zhang and Y. Liu, “Data imputation using least squares support vector machines in urban arterial streets,” IEEE Signal Process. Lett., vol. 16, no. 5, pp. 414–417, May 2009.

[18] J. Zhao and S. Sun, “High-order Gaussian process dynamical models for traffic flow prediction,” IEEE Trans. Intell. Transp. Syst., vol. 17, no. 7, pp. 2014–2019, Jul. 2016.

[19] H.-F. Yang, T. S. Dillon, and Y.-P.-P. Chen, “Optimized structure of the traffic flow forecasting model with a deep learning approach,” IEEE Trans. Neural Netw. Learn. Systems, vol. 28, no. 10, pp. 2371–2381, Oct. 2017.

[20] W. Huang, G. Song, H. Hong, and K. Xie, “Deep architecture for traffic flow prediction: Deep belief networks with multitask learning,” IEEE Trans. Intell. Transp. Syst., vol. 15, no. 5, pp. 2191–2201, Oct. 2014.

[21] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, “Traffic flow prediction with big data: A deep learning approach,” IEEE Trans. Intell. Transp. Syst., vol. 16, no. 2, pp. 1–9, 2nd Quart., 2014.

[22] S. Du, T. Li, X. Gong, Y. Yang, and S. J. Horng, “Traffic flow forecasting based on hybrid deep learning framework,” in Proc. 12th Int. Conf. Intell. Syst. Knowl. Eng. (ISKE), Nov. 2017, pp. 1–6.

[23] Y. Wu and H. Tan, “Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework,” 2016, arXiv:1612.01022 [Online]. Available: http://arxiv.org/abs/1612.01022

[24] J. Wang, Q. Gu, J. Wu, G. Liu, and Z. Xiong, “Traffic speed prediction and congestion source exploration: A deep learning method,” in Proc. IEEE 16th Int. Conf. Data Mining (ICDM), Dec. 2016, pp. 499–508.

[25] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, “Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction,” Sensors, vol. 17, no. 4, p. 818, Apr. 2017.

[26] A. Koesdwiady, R. Soua, and F. Karray, “Improving traffic flow prediction with weather information in connected cars: A deep learning approach,” IEEE Trans. Veh. Technol., vol. 65, no. 12, pp. 9508–9517, Dec. 2016.

[27] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic routing between capsules,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 3856–3866.

[28] Y. Kim, P. Wang, Y. Zhu, and L. Mihaylova, “A capsule network for traffic speed prediction in complex road networks,” in Proc. Sensor Data Fusion, Trends, Solutions, Appl. (SDF), Oct. 2018, pp. 1–6.

[29] W. Zhao, J. Ye, M. Yang, Z. Lei, S. Zhang, and Z. Zhao, “Investigating capsule networks with dynamic routing for text classification,” 2018, arXiv:1804.00538 [Online]. Available: http://arxiv.org/abs/1804.00538

[30] H. B. Huang, X. R. Huang, R. X. Li, T. C. Lim, and W. P. Ding, “Sound quality prediction of vehicle interior noise using deep belief networks,” Appl. Acoust., vol. 113, pp. 149–161, Dec. 2016.

[31] X. Ma, H. Yu, Y. Wang, and Y. Wang, “Large-scale transportation network congestion evolution prediction using deep learning theory,” PLoS ONE, vol. 10, no. 3, Mar. 2015, Art. no. e0119044.

[32] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, “Long short-term memory neural network for traffic speed prediction using remote microwave sensor data,” Transp. Res. C, Emerg. Technol., vol. 54, pp. 187–197, May 2015.