Research Article

Mass Rapid Transit System Passenger Traffic Forecast Using a Re-Sample Recurrent Neural Network

Rong Hu,¹ Yi-Chang Chiu ,² Chih-Wei Hsieh ,² Tang-Hsien Chang ,³ Xingsi Xue ,⁴ Fumin Zou,¹ and Lyuchao Liao¹

¹Fujian Province Key Laboratory of Automotive Electronics and Electric Drive, Fujian University of Technology, Fuzhou 350118, China

²Dept. of Civil and Architectural Engineering and Mechanics, The University of Arizona, Tucson, AZ 85721, USA

³School of Transportation, Fujian University of Technology, Fuzhou 350108, China

⁴Computer Science and Technology, Fujian University of Technology, Fuzhou 350108, China

Correspondence should be addressed to Yi-Chang Chiu; calmag@gmail.com

Received 4 December 2018; Revised 5 March 2019; Accepted 31 March 2019; Published 12 May 2019

Academic Editor: Guohui Zhang

Copyright © 2019 Rong Hu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In this study, we developed a model re-sample Recurrent Neural Network (RRNN) to forecast passenger traffic on Mass Rapid Transit Systems (MRT). The Recurrent Neural Network was applied to build a model to perform passenger traffic prediction, where the forecast task was transformed into a classification task. However, in this process, the training dataset usually ended up being imbalanced. To address this dataset imbalance, our research proposes re-sample Recurrent Neural Network. A case study of the California Mass Rapid Transit System revealed that the model introduced in this work could timely and effectively predict passenger traffic of MRT. The measurements of passenger traffic themselves were also studied and showed that the new method provided a good understanding of the level of passenger traffic and was able to achieve prediction accuracy upwards of 90% higher than standard tests. The development of this model adds value to the methodology of traffic applications by employing these Recurrent Neural Networks.

1. Introduction

Traffic congestion is a serious problem encountered in the implementation of MRT (Mass Rapid Transit) Systems. Delays can seriously affect people's quality of life and urban development, while overcrowding remains a top source of customer dissatisfaction. As the number of passengers choosing to use MRT as their major mode of transportation continues to increase, overcrowding during peak hours has become a common occurrence when ridership exceeds the capacity of the train. For this reason, it is very important to distribute congestion information to the public timely, such that they are able to rearrange their plans or departure time accordingly in order to avoid congestion. On the other hand, subway operators can also explore effective approaches to solving this congestion. They can create more standing room by removing seats in their existing fleets and can add additional train cars to allow them to run longer trains during peak periods. They can also consider adding enhanced

traction power and train control systems, as well as rail car storage space.

The increasing availability of data from system operators has created unique opportunities to predict congestion [1]. Predicting the crowdedness of a Rapid Transit System benefits the train operators and the public. Most existing work on predicting traffic has focused on predicting crowd flows [2–4]. For instance, Sun et al. [2] selected nonparametric regression as a prediction method to forecast pedestrian congestion. Zhang et al. [3] developed a real-time system based on Microsoft Azure Cloud to monitor and forecast crowd flow. Nicholas G. Polson et al. [4] developed a deep learning model to capture nonlinear, spatiotemporal effects and show how deep learning can provide precise, short-term traffic flow predictions. There are also other works that have focused on predicting traffic congestion, such as Min et al. [5], which proposed an adaptive, data-driven, real-time congestion prediction method which employed an adaptive K-means clustering method to identify different traffic patterns.

Ma et al. [6] extended deep learning theory into large-scale transportation. They utilized a deep Restricted Boltzmann Machine and Recurrent Neural Network architecture to model and predict traffic congestion evolution based on Global Positioning System (GPS) data from taxi. Deep Neural Networks can capture the nonlinearity of history data but are limited by the imbalance of the data which is a common phenomenon in the real world. In particular for the data-driven methods, the accuracy of their traffic pattern recognition is low due to data imbalance. In our previous work [7], we presented re-sample Deep Neural Networks for congestion prediction. Although it reported a nice forecasting result, it still has some room to improve in predicting. In this work, we employ deep learning Recurrent Neural Networks in our model to predict the passenger traffic of MRT. Passenger traffic is studied and indicated by several patterns according to congested level. As previously mentioned, the history data utilized is imbalanced, as the majority of the data implies normal conditions with little to no congestion. Only a very small fraction of the data shows severe congestion occurring during the peak times. If we use a deep learning model to represent this tremendously imbalanced data, it would always be overfitted by the large amount of normal data and fail to establish a good generalization on the unseen severe congestion data. Thus, it cannot predict the true congestion level in a timely manner. In order to solve this problem, we propose re-sample Recurrent Neural Network Model (RRNN) to effectively predict congestion levels. We use the traffic data from the Rapid Transit System of San Francisco in the US as a case study. These experiments show that the model's predictions were able to achieve an accuracy of more than 90% within 20 minutes.

The remainder of this paper is organized as follows: related works and literature review are given and discussed in Section 2. The definition of congestion level and a description of the data structure are introduced in Section 3. Section 4 discusses the modular re-sample Recurrent Neural Network used to predict passenger traffic patterns. We evaluate our results in Section 5 and summarize our research in relation to prior works in Section 6.

2. Literature Review

The Rapid Transit, or urban metro, System has been an extremely important component in urban infrastructures [8–10] and plays an essential role in the development of cities. According to statistics and prior studies, there are about 213 cities around the world that have deployed MRT Systems since the first line opened in 1863 in London [11]. On the one hand, MRT provides a certain level of convenience for people, but on the other hand, the crowdedness of MRT as a whole results in delays and heavily affects people's day-to-day lives. This crowdedness is always more commonplace during peak hours, when people are heading to or from work. This phenomenon has sparked a large amount of research focused on investigating this issue of overcrowding [12, 13]. In addition to this, many research efforts have also been made in predicting passenger flow [14, 15]. For example, [15] proposes a multipattern deep fusion technique constructed

by fusing deep belief networks. For each different pattern, a DBN is developed as a deep representation for passenger flow. This process ends up making the model much more complex. Some works focus on short-term metro passenger flow using parametric [16] and nonparametric techniques [17]. Generally, the application of parametric methods has some limitations because of a linear assumption within time lagged variables. Unlike the parametric models, nonparametric ones construct a nonlinear relationship between input and output without any prior knowledge. Some hybrid models integrating both parametric and nonparametric methods are designed to improve prediction accuracy [18, 19]. These models focus on short-term passenger flow prediction.

Our work will mainly focus on studying passenger flow pattern especially for congestion level prediction. Timely and accurate recognition of congestion can lower the negative impacts on a MTR (Mass Rapid Transit) System [20]. In order to more accurately describe traffic conditions, many scholars apply networks and fuzzy theories including the FCM algorithm, ANN algorithm, and DS-ANN algorithm [21].

Machine learning has also been widely applied and has shown considerable success in traffic pattern recognition [22]. Chen [23] used shallow neural networks for traffic applications. They applied a dynamic neural network with a single-hidden layer which used a Gaussian basis function as an activation unit. Zheng et al. [24] deployed multiple single-hidden layer networks to predict traffic within the next fifteen-minute time span. In their work, they used both a tanh activation and a Gaussian radial basis function. Cetiner [25] proposed a neural network model using the day of the week and the time of day as inputs. Lv et al. [17] demonstrated that deep learning can be effectively used for traffic forecasting. A stacked auto-encoder was applied to recognize spatiotemporal patterns in the traffic data. Zhao et al. [26] proposed a LSTM network to forecast traffic by considering spatiotemporal correlations in traffic systems via a two-dimensional network.

Although some efforts have been made to model spatiotemporal characteristic traffic flows in prior studies [4, 26], there have been no efforts to predict the passenger traffic patterns of MRT in a timely manner. Basically, deep learning allows us to design flexible network structures with multiple layers, and as such models built on deep learning perform better than traditional Neural Network Models. In recent years, deep learning has become a very popular technology, especially in dealing with image recognition and natural language processing [27]. A Recurrent Neural Network (RNN) is a type of deep learning module that has been applied to a wide range of applications including speech recognition [28] and text generation [29]. Some prior studies have also tried to use a RNN model for traffic accident prediction [30].

This work develops a re-sample RNN model to predict the passenger traffic patterns of MRT along with the spatiotemporal structure of passenger trip data. Generally, this training data is imbalanced due to congestion that only occurs during the peak period. However, the other 90% of the time it is seat available, without any congestion. If this kind of dataset

is directly used as input data to train the model, it is highly likely that it overfits for major patterns such as no congestion and fails to display a good representation of minor patterns like congestion, which are typically what researchers are interested in. In order to solve this problem, we propose a re-sample Recurrent Neural Network referred to as RRNN.

3. Description of the Data Structure and Passenger Traffic Patterns Definition

In this study, we focus on the MRT of San Francisco in the US. The dataset contains days ranging from January 1, 2017, to November 30, 2017. The data consists of three categories of variables: demand, supply, and day attributes. After testing numerous variables, 45 variables were included in the final model, categorized into three types. The first type is demand variables that include average segment passenger loads for the previous 28 days and events deemed likely to generate high demand. The second type is supply-related variables that include scheduled capacity and unscheduled capacity at each time interval of interest and schedule type. Scheduled capacity can be calculated through the train timetable of the operator of the MTR. The unscheduled capacity is estimated when the unscheduled trains are dispatched. The third type consists of a period, day of the week, month, week of the month, day type, raining day, and event indicator variables. Event indicators are identified by a list of special events, such as sporting events or entertainment events recorded by the operator of the MTR. The schedule and location of events are assumed to be known in advance. Each period is defined to be 20 minutes for the modeling process. The operating hours is from 4:00 am to midnight on weekdays, from 6:00 am to midnight on Saturday, and from 8:00 am to midnight on Sunday. The whole system every week has 7 days * 60 time segments * 92 links, and the number of rides in every segment in every station is recorded by operator. The output is the predicting result: severe, moderate, light, and normal. The training data includes 70% of the entire dataset, which translates to around 9 months of history data. The 2 remaining months of data are used as testing data. In total, the dataset consists of 833520 data samples. The output and input variable structure is listed in the Appendix.

According to common understanding, passenger traffic patterns are designed according to congestion level which is defined as the average amount of passengers per train car. According to this work, given 40 seats are available per car on MRT, the passenger traffic patterns are defined as 4 different patterns: congested, moderate, light, and normal, which are separated by three respective cutoff points: 120, 80, and 40. As such, if the number of passengers per car is less than 40, then congestion is normal, between 40 and 80 is light, between 80 and 120 is moderate, and over 120 is congested.

4. Module

Recurrent Neural Networks (RNNs) are neural networks with internal connections between hidden neurons and specifically designed feedback connections. The premise of this design is that human beings do not start their

thought process from scratch. The human mind has the ability to associate previous information with current events, a phenomenon called persistence of memory. However, traditional neural networks are unable to replicate this and end up ignoring previous information. Using a movie scene classifier as an example, a traditional neural network cannot utilize any previous scenes to predict the current one.

In contrast with traditional neural network, RNNs are networks with a loop that allows the network to retain that prior information. An RNN introduces a transition weight W to transfer information between time slots. RNNs process sequential input data once and update a vector state that contains past information about past events in the sequence. A neural network that takes input as a value of $X(t)$ and then outputs a value $Y(t)$ is shown in Figure 1 [31].

As shown in Figure 1, the first layer of the neural network is characterized by the function $Z(t) = X(t) \times W_{in}$, where $X(t)$ is the input sequence and $Z(t)$ is the state of the hidden neuron. The output is $Y(t) = Z(t) \times W_{out}$. $Z(t)$ can be updated with time, and t is a time series equal to 1, 2, 3, ... A more explicit architecture can be found in Figure 2. For the hidden layer, the temporally shared W_2 is learned. At the same time, the input layer weight W_1 and the output layer weight W_3 are also learned [31].

After preprocessing, the data has 45 attributes which include 11 months of data (2017.1.1-2017.11.30). The total data are split into two parts. The first part, about 9 months' worth of data (2017.1.1-2017.9.30), is used as training data, while the 2 remaining months' data is used as test data. The total number of training data samples is 625140. The components and the proportion of every component over the total training data are shown in Table 1 and Figure 3.

As shown in Table 1 and Figure 3, both the proportions of congestion level and the proportion of moderate level data are very small.

In order to speed up the training, minibatch gradient descent is used to split the training dataset into smaller batches that are used to calculate model error and update model coefficients. Minibatch gradient descent seeks to find a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. Thus, during the training, every input sequence into the RNN is a minibatch of stochastic samples of the training dataset. When the training dataset is imbalanced like our data, it is very difficult for RNNs to accurately gain an understanding of the minor data patterns. In order to solve this problem, we propose a re-sample method. First, we split our training data into 4 subdatasets with respect to their class labels. During the training, every minibatch size of input data re-samples from these 4 different subdatasets according to the proportion of every class' data, which is shown in Figure 4.

As shown in Figure 4, for every training step, the input data weight re-samples from different subdatasets. w_j ($j=1,2,3,4$) is the re-sample weight. Suppose the minibatch size is predefined as *minibatch* which is re-sampled using the following formula:

TABLE 1: The data components of the 4 passenger traffic patterns.

	Congestion	Moderate	Light	Normal
Counts(10000)	1.0976	1.4208	6.0553	53.9403
Proportion(%)	2	2.3	9.7	86

FIGURE 1: A RNN architecture.

$$\begin{aligned} \text{minibatch} \quad (x) &= \sum_{i=1}^n (w_i B(x_i)) \\ \text{s.t.} \quad \sum_{i=1}^n w_i &= 1 \end{aligned} \quad (1)$$

where B is a predefined minibatch number, w_i is the weight of the re-sample from the i th subdataset, and x_i is the instance from the i th subdataset.

The whole architecture of our RRNN (re-sample Recurrent Neural Network) is shown in Figure 5.

The calculations of nodes in Figure 5 are done via the following equations.

$$z(t) = \varnothing(W_{in}X(t) + Whz(t) + b_{in}) \quad (2)$$

$$o(t) = W_{out}z(t) + b_{out} \quad (3)$$

$$\hat{y}(t) = \sigma(o(t)) \quad (4)$$

W_{in} and W_{out} are the weights of the input output layer, respectively, and Wh is the weight between hidden neurons. $\varnothing$ is the activation function, which is tanh for this research, and b_{in} , b_{out} are the biases of the input and output layer, respectively. σ is also an activation function, which is Softmax in this instance. The above computation is the process of forward propagation. We use BPTT (Back Propagation Through Time) during back propagation, which is illustrated by the following equations.

Because there are losses during every time slot, let us assume that the loss L is

$$L = \sum_{t=1}^T L(t) \quad (5)$$

The gradient of W_{out} , b_{out} is calculated as formula (6) and (7):

$$\begin{aligned} \frac{\partial L}{\partial b_{out}} &= \sum_{t=1}^T \frac{\partial L(t)}{\partial b_{out}} = \sum_{t=1}^T \frac{\partial L(t)}{\partial o(t)} \cdot \frac{\partial o(t)}{\partial b_{out}} \\ &= \sum_{t=1}^T \hat{y}(t) - y(t) \end{aligned} \quad (6)$$

$$\begin{aligned} \frac{\partial L}{\partial W_{out}} &= \sum_{t=1}^T \frac{\partial L(t)}{\partial W_{out}} = \sum_{t=1}^T \frac{\partial L(t)}{\partial o(t)} \cdot \frac{\partial o(t)}{\partial W_{out}} \\ &= \sum_{t=1}^T (\hat{y}(t) - y(t)) (z(t)) \end{aligned} \quad (7)$$

where $\hat{y}(t)$ is the output at step t and y is the true label.

Let us define the gradient of the hidden state at time t as the following:

$$\Delta(t) = \frac{\partial L}{\partial z(t)} \quad (8)$$

Then the gradient of input weight W_{in} , input bias b_{in} , and hidden weight Wh is calculated as follows:

$$\frac{\partial L}{\partial W_{in}} = \sum_{t=1}^T \text{diag}(1 - (z(t))^2) \Delta(t) (z(t-1))^T \quad (9)$$

$$\frac{\partial L}{\partial b_{in}} = \sum_{t=1}^T \text{diag}(1 - (z(t))^2) \Delta(t) (x(t))^T \quad (10)$$

$$\frac{\partial L}{\partial Wh} = \sum_{t=1}^T \text{diag}(1 - (z(t))^2) \Delta(t) (x(t))^T \quad (11)$$

5. Evaluation

To validate the effectiveness and efficiency of the proposed model RRNN, we perform a case study of San Francisco in the US. Minimizing cross-entropy error is used as the optimization objective during the RRNN model training procedure. The cross-entropy error indicates the distance between the probability distributions of network outputs and target labels. The cross-entropy error is defined in the following equation [12]:

$$\text{Cross-entropy Err (CEE)} = -\sum_i \hat{y}_i \log(y_i) \quad (12)$$

where $\hat{y}_i$ is the predicted probability of a value of class i , and y_i is the true probability for that class.

A confusion matrix is the specific table layout that visualizes the performance of the model. Each row of the

FIGURE 2: A RNN architecture where all the weights in all the layers must be learned with time.

FIGURE 3: The counts and proportions of different congestion levels in the training data.

FIGURE 4: Re-samples from different subdatasets for every batch-size sample.

matrix represents the instances in a predicted class, while each column represents the instance in a true class [31].

In pattern recognition, precision and recall are indicators of the performance of the model. Precision (positive predictive value) is the fraction of the relevant instances among the

retrieved instances, and recall (sensitivity) is the fraction of relevant instances retrieved over the total amount of relevant instances:

$$\begin{aligned} \text{precision} &= \frac{tp}{tp + fp} \\ \text{recall} &= \frac{tp}{tp + fn} \end{aligned} \quad (13)$$

While tp is the true positive, fp is a false positive and fn is a false negative.

The F1 score is a measure of accuracy of the test. It combines the precision p and the recall r of the test to calculate a score. The F1 score is the harmonic average of the precision and recall. The hyperparameters in our model are determined through experiments.

5.1. Discussion of the Performance of the Model on 4 Congestion Levels. As mentioned previously in Section 3, the dataset consists of 833520 data points ranging from 01/01/2017 to 11/30/2017. In order to train and test the model, the dataset was split into two parts. The first part was cut at 09/30/2017, which is the range from 01/01/2017 to 09/30/2017, and was used as training data. The remainder was used as test data.

The weights of the re-sample of the model need to be determined with prior tests. For the RRNN input, the time step and element size should be predetermined. In our work, we reshape the 45 dimensional data items into a 9×5 matrix, set the time step as 5, the element size as 9, the iteration step as 12000, and the minibatch number as 1000. In order to determine the hidden layer size, we trained the model with different hidden layer sizes to discover that 250 is an appropriate value. The entropy-loss and accuracy are shown in Figure 6.

From Figure 6, the training accuracy is 86.1%, the test accuracy is 86.4%, and the minibatch loss is 0.341623 when the re-sample weights (congestion, moderate, light, normal) are 0.15, 0.15, 0.2, 0.5 and the hidden layer size is 250. From the time axis on the bottom of Figure 6, the model training time is about 5 minutes when the hidden size is 250 and iteration

FIGURE 5: The architecture of RRNN.

step is 12000. We also notice that the training accuracy and test accuracy are close, which implies that the model has good generalization.

In order to further validate the performance of our model, we tried different rates of re-sample weights to find the best weights for accurate predictions. The results are shown in Figure 7.

From Figure 7(a), when the weights of re-sample (which decide how many instances are sampled from the congested, moderate, light, and normal subdatasets) are set to 0.15, 0.15, 0.2, 0.5 with hidden size 200, the average test accuracy is 83.7%. More specifically, the precision of the congested level is 89% and the precision of the normal level is 92%. The normal level data is represented well by the model, because the fraction of training data is large. For the moderate and light levels, the precision is 70% and 71%, respectively. The average of precision, also referred to as the F1-score, is all 84%. From Figure 7(b), when the weights are set to average (0.25, 0.25, 0.25, 0.25) with hidden size 150, the model performs well as the precision of the congested level is 94% and the precision of the normal level is 89%. This shows that the model can effectively predict the congested and normal levels, which are the most important to operators and commuters. From Figure 7(c), when the weights of re-sample are set to 0.15, 0.15, 0.2, 0.5 and the hidden size is set to 250, the model also has a very good performance. It

shows that the model can effectively predict congestion where the precisions of the congested and normal levels are both higher than 90%. Also, the minimum of all the F1-scores is 0.76, which shows that the model can predict unseen data effectively. From Figure 7(c), having the model without re-sample although the average precision is very high (more than 90%), the recall is very low, which implies that the module failed to gain a good representation of the congestion level. As such, this module cannot be used to predict the congestion of passenger traffic. As shown in Figure 7(d), we set the weights to the same rate as the training data component, which is 86% normal data, 9.7% light level data, 2.3% moderate level data, and 2% congested data. The performance of the model was poor for the moderate level and the F1-score was very low, which shows that it has bad generalization.

The summary of the results of the model with different re-sample weights and hidden sizes is shown in Table 2.

From Table 2, the test accuracy represents the total accuracy of the test data. We notice that when the re-sample weights are set to (0.25, 0.25, 0.25, 0.25) with a hidden size of 200, the model performs better. Both the precision of severe congestion and the precision of the normal level are higher than 90%, and, at the same time, the performance of the moderate and light levels are also higher than other model. Furthermore, on average the recall of this model is better than others, which indicates that the model has a good

TABLE 2: The performance of models with different weights of resample rate (congestion, moderate, light, and normal) and different hidden sizes.

model	Test		precision(severe, moderate,light,seat available)				recall(severe, moderate,light,seat available)				F1-score(severe, moderate,light,seat available)			
	Accuracy		moderate		light,seat available		moderate		light,seat available		severe		moderate,light,seat available	
Weights(0.25,0.25,0.25,0.25)HiddenSize 200	0.792		0.88	0.69,	0.71,	0.92	0.88,	0.77,	0.70,	0.82	0.88,	0.73,	0.70,	0.87
Weights(0.2,0.2,0.2,0.4)HiddenSize 200	0.81		0.86,	0.72,	0.68,	0.90	0.81,	0.73,	0.69,	0.90	0.83,	0.72,	0.69,	0.90
Weights(0.15,0.15,0.2,0.5)HiddenSize 200	0.837		0.89,	0.70,	0.71,	0.92	0.80,	0.76,	0.69,	0.93	0.84,	0.73,	0.70,	0.92
Weights(0.02,0.023,0.097,0.086)HiddenSize 200	0.92		0.57,	0.56,	0.73,	0.95	0.40,	0.43,	0.63,	0.98	0.47,	0.49,	0.68,	0.97
Weights(0.2,0.2,0.2,0.4)HiddenSize 150	0.811		0.81,	0.70,	0.74,	0.90	0.83,	0.70,	0.68,	0.92	0.82,	0.70,	0.71,	0.92
Weights(0.15,0.15,0.2,0.5)HiddenSize 150	0.836		0.83,	0.68,	0.73,	0.94	0.79,	0.79,	0.74,	0.90	0.81,	0.73,	0.74,	0.92
Weights(0.25,0.25,0.25,0.25)HiddenSize 150	0.827		0.94,	0.72,	0.78,	0.89	0.84,	0.86,	0.70,	0.90	0.89,	0.78,	0.74,	0.90
Weights(0.02,0.023,0.097,0.86)HiddenSize 150	0.937		0.69,	0.68,	0.78,	0.96	0.45,	0.57,	0.68,	0.99	0.55,	0.62,	0.73,	0.97
Weights(0.2,0.2,0.2,0.4)HiddenSize 250	0.832		0.88	0.67,	0.74,	0.95	0.84,	0.81,	0.68,	0.92	0.86,	0.73,	0.71,	0.93
Weights(0.15,0.15,0.2,0.5)HiddenSize 250	0.86		0.91,	0.74,	0.78,	0.91	0.81,	0.78,	0.74,	0.95	0.80,	0.72,	0.79,	0.95
Weights(0.25,0.25,0.25,0.25)HiddenSize 250	0.791		0.82,	0.66,	0.79,	0.92	0.82,	0.76,	0.72,	0.88	0.82,	0.70,	0.75,	0.90
Weights(0.02,0.023,0.097,0.86)HiddenSize 250	0.93		0.79,	0.53,	0.71,	0.97	0.55,	0.43,	0.71,	0.98	0.65,	0.48,	0.71,	0.97
Without resample Hidden size 250	0.927		0.56,	0.78,	0.68,	0.96	0.62,	0.54,	0.67,	0.97	0.59,	0.64,	0.67,	0.97

FIGURE 6: The cross-entropy and accuracy when re-sample weights (congestion, moderate, light, normal) are set to (0.15,0.15,0.2,0.5) and the hidden size is 250.

generalization for unseen data. The model with the re-sample weights (0.86, 0.097, 0.023, 0.02) has the worst performance, as it cannot predict the crowdedness correctly. This weight rate is the exact rate of the distribution of the training data which consists of 86% normal data, 9.7% light data, 2.3% moderate data, and 2% congested data, which is basically equivalent to not having a re-sample. The model without resampling has a higher total accuracy, but it has bad recall on the congested and moderate levels, which implies that the model has bad generalization for unseen data. This essentially means that a re-sample on imbalanced data performs better than no re-sample at all for this model. We also notice that when the hidden size increases, the precision also slightly increases, but not obviously. In order to manifest more clearly, a comparison of congestion level prediction is given by Figure 8 among models with different weights.

According to Figure 8(a), the models are all with hidden size 250. It is noticed that the models with re-sample weights rate (0.15,0.15,0.2,0.5) has the best performance in precision and also well in recall and F1-score, the model with weights average rates has also much better performance than the model without re-sample. Even the model with re-sample weights rate (0.02,0.023,0.097,0.86) is better than the model without resampling. According the previous introduction,

this weights rate is the exact rate of four class dataset. The efficiency of the proposed weighted re-sample can also be certified by Figure 8(b). A comparison of F1-score among models with different weights is given by Figure 8(b). According to Figure 8(b) the model without resampling works worst when it is predicting congestion level and light level. The models with re-sample weights rate (0.2,0.2,0.2,0.4) work better than others when it is predicting congestion level. The model with re-sample average weights can also get a good performance on congestion level yet a little bad performance on normal. This is because that the training data size of normal level is big, when raising the rate of re-sample weights, the model works better on predicting normal level.

5.2. Discussion of the Performance of the Model on 3 Congestion Levels. Another notable case from the confusion matrices in Figure 7 is that there was more confusion in predicting the moderate and light levels. For instance, from Figure 7(b), there are 29/250 instances of moderate misclassified to light and 53/250 of light misclassified to moderate. We can also see from Table 2 that the model has lower precision and recall on these two levels. We think the distinction between light and moderate is not sufficient for the model to gain a good representation of them. Based on this analysis, we redefine

[[120 15 1 14] [10 114 25 1] [1 33 139 27] [4 0 32 464]]				
	precision	recall	f1-score	support
congestion	0.89	0.80	0.84	150
moderate	0.70	0.76	0.73	150
light	0.71	0.69	0.70	200
normal	0.92	0.93	0.92	500
avg / total	0.84	0.84	0.84	1000

(a) Weight (congestion, moderate, light, normal) is 0.15,0.15,0.2,0.5 with hidden size 200

[[211 28 4 7] [7 214 29 0] [0 53 176 21] [6 1 17 226]]				
	precision	recall	f1-score	support
congestion	0.94	0.84	0.89	250
moderate	0.72	0.86	0.78	250
light	0.78	0.70	0.74	250
normal	0.89	0.90	0.90	250
avg / total	0.83	0.83	0.83	1000

(b) Weight (congestion, moderate, light, normal) is 0.25,0.25,0.25,0.25 with hidden size 150

[[121 16 1 12] [5 117 24 4] [0 23 149 28] [7 2 18 473]]				
	precision	recall	f1-score	support
congestion	0.91	0.81	0.86	150
moderate	0.74	0.78	0.76	150
light	0.78	0.74	0.76	200
normal	0.91	0.95	0.93	500
avg / total	0.86	0.86	0.86	1000

(c) Weight (congestion, moderate, light, normal) is 0.15,0.15,0.2,0.5 with hidden size 250

Test Accuracy: 93.0 [[11 6 0 3] [0 10 11 2] [0 3 69 25] [3 0 17 840]]				
	precision	recall	f1-score	support
congestion	0.79	0.55	0.65	20
moderate	0.53	0.43	0.48	23
light	0.71	0.71	0.71	97
normal	0.97	0.98	0.97	860
avg / total	0.93	0.93	0.93	1000

(d) Weight (congestion, moderate, light, normal) is 0.02,0.023,0.97,0.86 with hidden size 250

[[10 2 0 4] [0 14 10 2] [2 2 61 26] [6 0 19 842]]				
	precision	recall	f1-score	support
congestion	0.56	0.62	0.59	16
moderate	0.78	0.54	0.64	26
light	0.68	0.67	0.67	91
normal	0.96	0.97	0.97	867
avg / total	0.93	0.93	0.93	1000

(e) Without re-sample with hidden size 250

FIGURE 7: The confusion matrices with different re-sample weights and different hidden sizes. The confusion matrices are set with rows from congestion to normal from top to bottom.

the passenger traffic pattern into 3 patterns (congestion, moderate, and normal). We relabel our congestion levels into 3 classes including normal, light congestion, and congestion. The moderate and light levels are combined into one level named moderate. Thus, the training data consists of 86% normal, 12% moderate, and 2% congestion data. The cross-entropy is shown in Figure 9, while the confusion matrix and corresponding performance are shown in Figure 10.

From Figure 9, the weights of re-sample (congestion, moderate, normal) are set to the average of the minibatch size. It shows that the training accuracy and the test accuracy of the model are 91.7% and 91%, respectively. The accuracy and cross-entropy of the training and test are very close, such that the model has good robustness. From Figure 10, it shows that the precision and recall of the congested and normal levels are higher than 90%, while the precision of moderate is 86% and the recall is 91%, which means that the model can effectively predict unseen data. From Figure 9, it also shows that the training time is about 8 minutes. In order to decide the appropriate weights of re-sample and the hidden size, we tried different weight rates, and the summary of those experiments can be found in Table 3. From Table 3, we compared the

different weight rates and hidden sizes. When the weights are set to an average of 0.33 congested, 0.33 moderate, and 0.34 normal and the hidden size is 250 (the bold row), the model has better performance. The other two models (the italic row) have also reached a similar level of performance. In essence, when the hidden size is increased, the model performance improves slightly, but not obviously. However, when the hidden size is under 150, the model performance decreases significantly.

5.3. Discussion of the Influence of the Parameter Selection in Model. In this study, the data consists of three categories of variables: demand, supply, and day attributes. After testing numerous variables, 45 were included in the final model, categorized into three types. The first type is demand variables. The second type is supply-related variables. The third type consists of a period, day of the week, month, week of the month, day type, raining day, and event indicator variables. In our work, we reshape the 45 dimensional data items into a 9*5 matrix, set the time step as 5, the element size as 9. We did not deliberately arrange the sequence of the variables. According to the attributes of RRN, here there are 5

TABLE 3: The performance of models with different weights of resample rate (congestion, moderate, and normal) and different hidden sizes.

model	Test Accuracy	precision(congestion, moderate,normal)		recall((congestion, moderate,normal)		FIscore(congestion, moderate, normal)	
Weights(0.25,0.25,0.5)HiddenSize 200	0.9	0.89,	0.85,	0.93	0.92,	0.91,	0.84,
Weights(0.3,0.3,0.4) HiddenSize 200	0.907	0.92,	0.85,	0.95	0.94,	0.93,	0.87,
Weights(0.33,0.33,0.34) HiddenSize 200	0.887	0.89,	0.87,	0.91	0.92,	0.90,	0.86,
Weights(0.02,0.12,0.86)HiddenSize 200	0.947	0.61,	0.85,	0.97	0.55,	0.58,	0.82,
Weights(0.25,0.25,0.5)HiddenSize 150	0.89	0.81,	0.86,	0.95	0.96,	0.88,	0.84,
Weights(0.3,0.3,0.4) HiddenSize 150	0.903	0.93,	0.85,	0.93	0.88,	0.90,	0.87,
Weights(0.33,0.33,0.34) HiddenSize 150	0.886	0.91,	0.85,	0.89	0.88,	0.90,	0.86,
Weights(0.02,0.12,0.86)HiddenSize 150	0.941	0.75,	0.79,	0.96	0.45,	0.56,	0.80,
Weights(0.25,0.25,0.5)HiddenSize 250	0.91	0.91	0.85,	0.89	0.94,	0.91,	0.86,
Weights(0.3,0.3,0.4)HiddenSize 250	0.898	0.95,	0.84,	0.81	0.87,	0.91,	0.86,
Weights(0.33,0.33,0.34)HiddenSize 250	0.91	0.95,	0.86,	0.92	0.92,	0.93,	0.89,
Weights(0.02,0.12,0.86)HiddenSize 250	0.954	0.82,	0.81,	0.98	0.70,	0.76,	0.84,
Without resample Hidden size 250	0.938	0.50,	0.84,	0.97	0.09,	0.15,	0.88,

FIGURE 9: The accuracy and cross-entropy of the model with the weights of re-sample set to average and the hidden size set to 250.

TABLE 4: A comparison between two models with different sequence of time step.

	Test Accuracy	Precision (Congestion, Moderate, Light, Normal)				Recall (Congestion, Moderate, Light, Normal)				F1-score (Congestion, Moderate, Light, Normal)			
model 1	83.9	85	71	72	93	79	75	74	92	82	73	73	92
model 2	83.7	89	70	71	92	80	76	69	93	84	73	70	92

```
Test Accuracy: 91.0
[[302 23 5]
 [ 8 301 21]
 [ 9 24 307]]
```

	precision	recall	f1-score	support
congestion	0.95	0.92	0.93	330
moderate	0.86	0.91	0.89	330
normal	0.92	0.90	0.91	340
avg / total	0.91	0.91	0.91	1000

FIGURE 10: Confusion matrix and the resulting report with re-sample. Weights set to average and hidden size set to 250.

has been found that weight re-sampling from imbalanced training data can significantly promote the performance of

the prediction model. Third, the development of the model applying Recurrent Neural Networks in this paper adds value to the development of methodologies for further studies on the utility performance of traffic domains.

Appendix

See Table 5.

Data Availability

The data we used is not allowed to be published without the permission of company.

TABLE 5: The output and input variable list.

Column name		Output variable	
V1		Severe	
V2		Moderate	
V3		Light	
V4		Normal	
Column name		Input variable	
c_sch		scheduled capacity	
c_unsch		unscheduled capacity	
p_vol		average passenger loads for the segment for day d that is within the previous 28 days.	
rain		Raining day variable for day d, if rains, r=1 otherwise 0.	
Coliseum		Event variable (0 / 1)	
Fremont		Event variable (0 / 1)	
Giants		Event variable (0 / 1)	
Berkeley		Event variable (0 / 1)	
Hardly		Event variable (0 / 1)	
Autoshow		Event variable (0 / 1)	
Dreamforce		Event variable (0 / 1)	
ArtMurmur		Event variable (0 / 1)	
holiday		h=1 if day d is a holiday, otherwise, h = 0	
V14		schedule_type:WE	
V15		schedule_type:SA	
V16		schedule_type:SU	
V17		Month:January	
V18		Month:February	
V19		Month:March	
V20		Month:April	
V21		Month: May	
V22		Month:June	
V23		Month:July	
V24		Month:August	
V25		Month:September	
V26		Month:October	
V27		Month:November	
V28		Month:December	
V29		day of week:Sunday	
V30		day of week:Monday	
V31		day of week:Tuesday	
V32		day of week:Wednesday	
V33		day of week:Thursday	
V34		day of week:Friday	
V35		day of week:Saturday	
V36	month of week		1
V37	month of week		2
V38	month of week		3
V39	month of week		4
V40	Time period dummy variable		5am to 8am
V41			8am to 11am
V42			11am to 2pm
V43			2pm to 5pm
V44			5pm to 8pm
V45			8pm to 11pm

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This research is funded by the Fujian Provincial Department of Science and Technology (Grant No. 2017J01729) and the China Scholarship Council (Grant No. 201709360005).

References

- [1] R. Hu and Y. Xia, "Traffic condition recognition based on vehicle trajectory big data," *Journal of Internet Technology*, vol. 18, no. 7, pp. 1587–1596, 2017.
- [2] Y. Sun, G. Zhang, and H. Yin, "Passenger flow prediction of subway transfer stations based on nonparametric regression model," *Discrete Dynamics in Nature and Society*, vol. 2014, Article ID 397154, 8 pages, 2014.
- [3] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, and T. Li, "Predicting citywide crowd flows using deep spatio-temporal residual networks," *Artificial Intelligence*, vol. 259, pp. 147–166, 2018.
- [4] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," *Transportation Research Part C: Emerging Technologies*, vol. 79, pp. 1–17, 2017.
- [5] W. Min and L. Wynter, "Real-time road traffic prediction with spatio-temporal correlations," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 4, pp. 606–616, 2011.
- [6] X. Ma, H. Yu, Y. Wang, Y. Wang, and J. Gomez-Gardenes, "Large-scale transportation network congestion evolution prediction using deep learning theory," *PLoS ONE*, vol. 10, no. 3, Article ID e0119044, 2015.
- [7] R. Hu, "Congestion prediction on rapid transit system based on weighted resample deep neural network," in *Proceedings of the Euro-China Conference on Intelligent Data Analysis and Applications*, pp. 586–593, Springer, Cham, Switzerland, 2018.
- [8] L. Jiao, L. Shen, C. Shuai, Y. Tan, and B. He, "Measuring crowdedness between adjacent stations in an urban metro system: A Chinese case study," *Sustainability*, vol. 9, no. 12, p. 2325, 2017.
- [9] B. P. Y. Loo and A. H. T. Cheng, "Are there useful yardsticks of population size and income level for building metro systems? Some worldwide evidence," *Cities*, vol. 27, no. 5, pp. 299–306, 2010.
- [10] C. E. S. De Andrade and M. D. A. D'Agosto, "The role of rail transit systems in reducing energy and carbon dioxide emissions: The case of the city of Rio de Janeiro," *Sustainability*, vol. 8, no. 150, 2016.
- [11] "Metrobits," <http://mic-ro.com/metro/> (accessed on 7/3/2018).
- [12] H. Kim, S. Kwon, S. K. Wu, and K. Sohn, "Why do passengers choose a specific car of a metro train during the morning peak hours?" *Transportation Research Part A: Policy and Practice*, vol. 61, pp. 249–258, 2014.
- [13] K. M. Kim, S.-P. Hong, S.-J. Ko, and D. Kim, "Does crowding affect the path choice of metro passengers?" *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 292–304, 2015.
- [14] L. Liu and R.-C. Chen, "A novel passenger flow prediction model using deep learning methods," *Transportation Research Part C: Emerging Technologies*, vol. 84, pp. 74–91, 2017.
- [15] Y. Bai, Z. Sun, B. Zeng, J. Deng, and C. Li, "A multi-pattern deep fusion model for short-term bus passenger flow forecasting," *Applied Soft Computing*, vol. 58, pp. 669–680, 2017.
- [16] L. Du, S. Peeta, and Y. H. Kim, "An adaptive information fusion model to predict the short-term link travel time distribution in dynamic traffic networks," *Transportation Research Part B: Methodological*, vol. 46, no. 1, pp. 235–252, 2012.
- [17] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: a deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.
- [18] G. Xie, S. Wang, and K. K. Lai, "Short-term forecasting of air passenger by using hybrid seasonal decomposition and least squares support vector regression approaches," *Journal of Air Transport Management*, vol. 37, pp. 20–26, 2017.
- [19] N. Glišović, M. Milenković, N. Bojović, L. Švadlenka, and Z. Avramović, "A hybrid model for forecasting the volume of passenger flows on Serbian railways," *Operational Research*, vol. 16, no. 2, pp. 271–285, 2016.
- [20] X. Li, Z. Lv, J. Hu et al., "Traffic management and forecasting system based on 3d gis," in *Proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 991–998, IEEE, 2015.
- [21] C. N. Babu and B. E. Reddy, "A moving-average filter based hybrid ARIMA-ANN model for forecasting time series data," *Applied Soft Computing*, vol. 23, pp. 27–38, 2014.
- [22] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural Networks*, vol. 32, pp. 333–338, 2012.
- [23] H. Chen and S. Grant-Muller, "Use of sequential learning for short-term traffic flow forecasting," *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 5, pp. 319–336, 2001.
- [24] W. Zheng, D. H. Lee, and Q. Shi, "Short-term freeway traffic flow prediction: bayesian combined neural network approach," *Journal of Transportation Engineering*, vol. 132, no. 2, pp. 114–121, 2006.
- [25] B. Gültekin Çetiner, M. Sari, and O. Borat, "A neural network based traffic-flow prediction model," *Mathematical and Computational Applications*, vol. 15, no. 2, pp. 269–278, 2010.
- [26] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. Liu, "LSTM network: a deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems*, vol. 11, no. 2, pp. 68–75, 2017.
- [27] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, pp. 160–167, ACM, 2008.
- [28] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech And Signal Processing (Icassp)*, pp. 6645–6649, IEEE, 2013.
- [29] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1017–1024, 2011.
- [30] M. I. Sameen and B. Pradhan, "Severity prediction of traffic accidents with recurrent neural networks," *Applied Sciences (Switzerland)*, vol. 7, no. 6, 2017.
- [31] G. Zaccane, M. R. Karim, and A. Menshawy, *Deep Learning with TensorFlow*, Packt Publishing Ltd, 2017.