Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 9488369, 14 pages
https://doi.org/10.1155/2021/9488369

Research Article

Short-Term IoT Data Forecast of Urban Public Bicycle Based on the DBSCAN-TCN Model for Social Governance

Dazhou Li,[1] **Chuan Lin**,[2] **Wei Gao**,[1] **Guangbao Yu**,[1] **Jian Gao**,[3] **and Wenbo Xia**[4]

[1]*College of Computer Science and Technology, Shenyang University of Chemical Technology, Shenyang 110016, China*
[2]*Software College, Northeastern University, Shenyang 110819, China*
[3]*Shenyang University of Technology, Shenyang 110870, China*
[4]*Liaoning Technical University, Huludao 123032, China*

Correspondence should be addressed to Jian Gao; gaojian_sut@foxmail.com

Received 28 June 2021; Revised 9 August 2021; Accepted 27 October 2021; Published 30 November 2021

Academic Editor: José A. García-Naya

Internet of Things will play a vital role in the public transport systems to achieve the concepts of smart cities, urban brains, etc., by mining continuously generated data from sensors deployed in public transportation. In this sense, smart cities applied artificial intelligence techniques to offload data for social governance. Bicycle sharing is the last mile of urban transport. The number of the bike in the sharing stations, to be rented in future periods, is predicted to get the vehicles ready for deployment. It is an important tool for the implementation of smart cities using artificial intelligence technologies. We propose a DBSCAN-TCN model for predicting the number of rentals at shared bicycle stations. The proposed model first clusters all shared bicycle stations using the DBSCAN clustering algorithm. Based on the results of the clustering, the data on the number of shared bicycle rentals are fed into a TCN neural network. The TCN neural network structure is optimized. The effects of convolution kernel size and Dropout rate on the model performance are discussed. Finally, the proposed DBSCAN-TCN model is compared with the LSTM model, Kalman filtering model, and autoregressive moving average model. Through experimental validation, the proposed DBSCAN-TCN model outperforms the traditional three models in terms of two metrics, root mean squared logarithmic error, and error rate, in terms of prediction performance.

1. Introduction

Smart cities employ technology and data to increase efficiencies, economic development, sustainability, and life quality for citizens in urban areas. Inevitably, clean technologies promote smart city development including energy, transportation, and health [1–3]. In this sense, smart cities present themselves as a viable solution to aggregate public resources, human capital, social capital and information, and communication technologies, to promote sustainable development [4]. For instance, the information gathered by the advancements of the intelligent transportation systems are progressively intricate and are portrayed by heterogeneous devices, huge volume, mistakes in spatial and transient procedures, and continuous necessities of real-time processing [5, 6]. Various countries throughout the world have started their efforts in designing and implementing smart cities [7]. This has led to the concept of smart cities where Information Communication and Technology are merged with the existing traditional infrastructure of a city, which is playing a vital role in policy design, decision, implementation, and ultimate productive services [8, 9].

The Internet of Things (IoTs) and artificial intelligence (AI) are two cornerstone technologies enabling the smart city concept, which are fusing into an organic whole in recent years. Some particular joint points where IoTs meet AI are intelligent IoT devices, smart sensing boosted by AI, and IoT big data mining with AI [10, 11]. The new Internet of Things paradigm and architecture allows one to rethink the way smart city infrastructures are designed and managed [12]. In the densely populated IoT applications, the sensing range of the nodes might overlap frequently [13]. The world will be populated by billions of connected IoT devices that will be placed in our homes, cities, vehicles, and industries

[14]. The novel IoT devices collect cyber-physical data and provide information on the environment [15]. The IoTs enable a smart city to power and monitor multiple geographically distributed nodes to support a range of applications across various domains such as energy and resource management, intelligent transport systems, and E-health to name a few [16]. Systems get smarter with computing capabilities, especially in the form of IoT devices [17].

In a variety of smart cities, AI has been widely deployed, yielding numbers of revolutionary applications and services that are primarily driven by techniques for data offloading for urban IoT [18, 19]. The growing urbanization coupled with the high demands of daily commute for working professionals has increased the popularity of public transport systems [20, 21]. In urban public transportation systems, IoTs are deployed in major cities of both developed and developing countries [22]. The huge growth of IoT devices and different characteristics in the traffic patterns have brought attention to traffic methods to address various raised issues in IoT applications [23].

Density-Based Spatial Clustering with Noise Applications (DBSCAN) is an unsupervised ML clustering algorithm. Unsupervised means that it does not use prelabeled targets to cluster data points. Clustering is the attempt to group similar data points into manually determined groups or clusters. It is an alternative to popular clustering algorithms such as K-Means and hierarchical clustering. The core idea of density clustering is to use the number of points in the neighborhood of a data object as the density of data points, which are then connected by density and added to the clusters adjacent to them, completing the construction of clustered clusters. It allows clusters made up of irregularly shaped data objects to be found and noise points to be discovered.

From the perspective of graph theory, the node and node relationship of the graph is used to represent the information of the dataset, each data object corresponds to a node in the graph, and the connection between two points represents that the data objects are connected to each other. The data objects in the same connected branch are more similar, and the data objects in different connected branches are less similar. For nonnumerical objects, a concept similarity measure can be used based on the concept represented by the object. The more common or similar properties exist between concepts, the more similar the objects are to each other, at which point the cluster is a collection of objects with some common properties. The DBSCAN is more commonly used in text clustering and WEB data mining.

2. Related Work

At this stage, there has been a great deal of research in the field of bicycle sharing, with Bonilla-Alicea et al. [24]. O'Brienet al. proposed a bicycle sharing system to reduce environmental pollution [25]. Kadri et al. investigated the problem of bike-sharing scheduling in a static mode bike-sharing system [26]. Erdoğan et al. proposed an exact algorithm, calculated accurate algorithms, and tested the benchmark examples [27].

For the vehicle rebalancing problem, in addition to manual bike scheduling, operators and governments encouraged the use of two-way incentives [28]. Zhang et al. proposed a forecasting method that considers bike inventory forecasts and user arrivals in a unified way [29]. Faghih-Imani and Eluru proposed destination forecasting with a polynomial model [30]. Zhang et al. proposed two new regression-based inference models to predict potential travel destinations [31]. Froehlich et al. [32] and Kaltenbrunner et al. [33] proposed a spatio-temporal analysis of bike-sharing stations in Barcelona using a clustering approach.

Vogel et al. conducted a predictive analysis of bike-sharing usage using a time-series approach [34]. Yoon et al. proposed an improved ARMA-based predictive model for predicting the number of shared bikes within bike-sharing stations [35]. Noland et al. investigated how population size and employment density can affect bike-sharing usage [36]. Gebhart and Noland investigated the effect of weather factors that can affect bike-sharing use and the duration of bike-sharing rides [37]. Borgnat et al. viewed bike-sharing systems as dynamic networks and conducted a spatial analysis of the dynamic networks to derive the spatial distribution patterns of the networks [38]. Campbell et al. noted that temperature, precipitation, inclement weather, and air quality all affect bike-sharing through a survey of bike-sharing projects in Beijing [39]. Kaspi et al. propose a Bayesian estimation model and use it to estimate the number of unavailable bikes in a bike-sharing station [40]. The assumption that bicycle failure is a binary property is a limitation of the proposed model. Since some bicycles need to be maintained and most of the bicycles are still be rented, the proposed model is not valid under these conditions. The long-term transaction history of the needs of the bicycle is considered to solve the problem.

Traditional forecasting studies on bike-sharing rentals have generally only predicted the number of rentals at individual stations. The impact of other nearby station rentals on the rentals of the single station under study is not taken into account in the process of predicting the rentals of a single station. The disadvantage not only reduces the accuracy of the forecast but also makes it difficult to place shared bikes. It can lead to an excess of shared bikes remaining at some stations. Simultaneously, there is a lack of shared bikes at some of the stations. It is a very irrational management method. A lot of time, material, and human resources are wasted. It harms the development of shared bikes.

Based on the above reasons, according to the geographical location of the shared bicycle stations, we use the DBSCAN clustering algorithm to cluster the shared bicycle stations. The clustering results of the shared bicycle stations are fed into a temporal convolutional network (TCN) to achieve the demand forecasting for the rental volume of multiple shared bicycle stations. Finally, the DBSCAN-TCN model is evaluated using error rate (ER) and root mean squared logarithmic error (RMLSE) as evaluation metrics.

3. Analytical Model of DBSCAN-TCN

Both LSTM and RNN are very effective methods in the field of time series prediction. However, both of them have high requirements on the length of the input sequence during

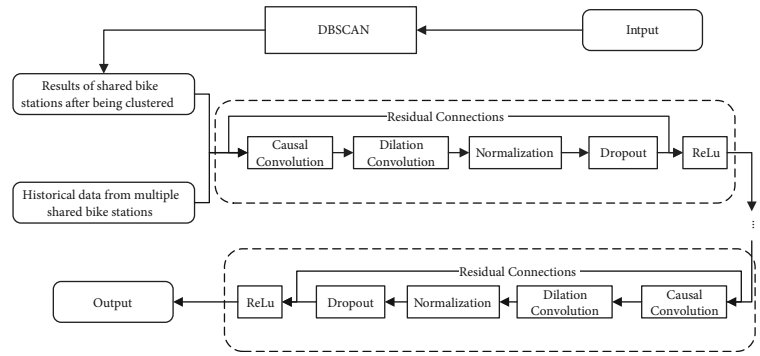

Figure 1: The architecture of the proposed DBSCAN-TCN model.

the computation. Also, the sequences they deal with are both one-dimensional. Long short-term memory (LSTM) and recurrent neural network (RNN) are not suitable when multiple time series are correlated and are considered as a whole. Figure 1 shows the architecture of the DBSCAN-TCN model proposed in the paper, which clusters that the bike-sharing stations in the geographic space are divided into different station clusters by the DBSCAN algorithm. Then, 48 hours of historical data from multiple bike-sharing stations belonging to the same station cluster are fed into the TCN. The DBSCAN-TCN model is shown in Figure 1.

3.1. DBSCAN Clustering of Public Bike-Sharing Stations. Compared to DBSCAN, K-Means is particularly susceptible to outliers. When the algorithm traverses the center of mass, outliers have a significant effect on how the center of mass moves before stability and convergence are achieved. In addition, K-Means suffers from the problem of clustering data exactly, when cluster sizes and densities vary. K-Means can only be applied to spherical clusters. If the data is not spherical, its accuracy suffers. Finally, K-Means requires that the number of clusters, which wishes to find be chosen first. On the other hand, DBSCAN does not require the number of clusters to be specified. It not only avoids outliers but determines that it works very well with clusters of any shape and size. DBSCAN has no center of mass. Clustering clusters are formed by joining neighboring points together.

In this paper, the DBSCAN clustering algorithm is chosen to cluster bike-sharing stations within a certain geographic space. The geographical information of the shared bicycle stations, i.e., latitude and longitude, is used as input to the DBSCAN clustering algorithm. According to the DBSCAN clustering rules, the clustering results of the shared bicycle stations within a certain geographic space are used as the output generated by the algorithm. To make better use of the DBSCAN algorithm, two important parameters of the DBSCAN clustering algorithm, *Eps* and *MmPtS*, need to be discussed. The parameters *Eps* and *MmPtS* are used to describe the closeness of the distribution of the neighborhood samples and the neighborhood distance threshold for a given sample, respectively.

Eps is the maximum radius of the community. If the distance of the data points from each other is less than or equal to the specified Eps, then they will be in the same class. It means that DBSCAN uses Eps to determine whether two points are similar and belong to the same class. Larger Eps will produce larger clusters, and smaller Eps will build smaller clusters. In general, the smaller value is chosen. Because only a very small percentage of the data points are within a distance of each other. However, if it is too small, the clusters will be split smaller. Step 2 of the DBSCAN algorithm was demonstrated to determine the best Eps.

Within the radius of a neighborhood, some MmPtS neighbors are considered to be a cluster. The initial points are contained in the MmPtS. A lower MmPtS helps the algorithm to build more clusters with more noise or outliers. A higher MmPtS will ensure more robust clusters. However, if the clusters are too large, the smaller clusters will be merged into the larger ones. In this paper, MmPtS is set to be greater than or equal to the dimensionality of the dataset. We multiply the number of dimensions of the features by two to determine their MmPtS values.

Assume that the dataset is $D = (x_1, x_2, \cdots, x_m)$. The density description of the DBSCAN clustering algorithm is defined as follows.

(1) *Eps Neighborhood*. For $x_j \in D$, the *Eps* neighborhood contains the subset of samples in the sample set D whose distance from is less than or equal to *Eps*, i.e., $N\epsilon(x_j) = \{x_i \in D \mid \text{distance}(x_i, x_j) \leq Eps\}$. The number of this subsample set is denoted as $|N\epsilon(x_j)|$

(2) *Core Objects*. For any sample $x_j \in D$, x_j is a core object, if its *Eps* neighborhood corresponds to $N\epsilon(x_j)$ containing at least *MinPts* samples, i.e., $|N\epsilon(x_j)| \geq MinPts$

(3) *Density Direct*. If x_i lies in the x_j neighborhood of *Eps* and is a core object, then it is said to be density direct from x_i to x_j. Note that the converse is not necessarily true. The density direct from x_j to x_i can not be confirmed unless and until x_i is also a core object

(4) *Density Reachable*. If $p_1, p_2, \cdots, p_T$ satisfies $p_1 = x_i$, $p_T = x_j$, and p_{t+1} directly reached p_t, it is said to be density reachable from x_i to x_j. Therefore, density reachability satisfies transmissibility. In this case, the transfer samples in the sequence $p_1, p_2, \cdots, p_T$ are

all core objects, as only the core objects can make the other samples' density reachable. It should be noted that density reachability also does not satisfy symmetry. It can be derived from the asymmetry of the density reach

(5) *Density Connected*. For x_i and x_j, if a sample of core objects x_k exists such that both x_i and x_j are accessible by density reachable from x_k, then x_i and x_j are said to be density connected. Density connectivity satisfies symmetry

The DBSCAN algorithm is calculated as shown below:

Input: sample set is $D=(x_1, x_2, \ldots, x_m)$, neighbourhood parameters (E, $MinPts$).

Output: Cluster division C.

Computational steps are as follows:

Step 1. The core set of objects $\Omega = \varnothing$, the cluster division $C = \varnothing$, the set of unvisited samples $\Gamma = D$, and the number of clusters $k = 0$ are initialized.

Step 2. For $j = 1, 2, \cdots, m$, all core objects are found according to the following steps. The subsample set $N\epsilon(x_j)$ of the E neighborhood of the sample is found by measuring the distance. If the number of samples in the subsample set satisfies $|N\epsilon(x_j)| \geq MinPts$, sample x_j is added to the sample set $\Omega = \Omega \cup \{x_j\}$ of the core objects.

Step 3. If the core set of objects is empty $\Omega = \varnothing$, the clustering algorithm ends. Otherwise, the algorithm proceeds to Step 4.

Step 4. A core object o is randomly selected in the core object set Ω. The current cluster core object queue $\Omega_{cur} = \{o\}$, the category ordinal number $k = k + 1$, and the current cluster sample set $C_k = \{o\}$ are initialized. The set of unvisited samples $\Gamma = \Gamma - \{o\}$ is updated.

Step 5. If the current cluster core object queue is empty and $\Omega_{cur} = \varnothing$, the current cluster C_k is generated. The cluster partition $C = \{C_1, C_2, \cdots, C_k\}$ and the set of core objects $\Omega = \Omega - C_k$ are updated. Otherwise, only the set of core objects $\Omega = \Omega - C_k$ is updated.

Step 6. A core object o' is removed from the current cluster core object queue Ω_{cur}. The set of all E neighborhood subsamples $N\epsilon(o')$ is found according to the neighborhood distance threshold E. Let $\Delta = N\epsilon(o') \cap \Gamma$. The current set of cluster samples $C_k = C_k \cup \Delta$, the set of unvisited samples $\Gamma = \Gamma - \Delta$, and $\Omega_{cur} = \Omega_{cur} \cup (\Delta \cap \Omega) - o'$ are updated. The algorithm moves to Step 5.

Output: The set of cluster divisions $C = \{C_1, C_2, ..., C_k\}$.

The data for Divvy Bike Share Chicago was processed according to the DBSCAN algorithm above [41]. The names of bike-share stations in the city of Chicago are typically long. To facilitate the recording of information about these stations, stations were described by the ID of each station. After DBSCAN clustering, seven cluster representatives of the clusters are given in Table 1. These seven clusters contain two bicycle stations, thirteen bicycle stations, four bicycle stations, twenty bicycle stations, eight bicycle stations, ten bicycle stations, and seven bicycle stations, respectively. The first cluster has only two cycle stations because of its remote location and low footfall. It is the cluster with the fewest bicycle stations of the site clusters. Cluster 4 has twenty bicycle stations, which is the cluster with the most bicycle stations. It is in the heart of the city of Chicago and has a high volume of foot traffic.

Table 1: The 7 typical station clusters and the contained stations after processing by the DBSCAN clustering.

Cluster number	The ID of the shared bike station
1	427, 413
2	47, 52, 35, 26, 81, 180, 197, 110, 211, 111, 43, 31, 177
3	260, 502, 290, 123
4	77, 288, 365, 60, 300, 138, 195, 47, 112, 225, 346, 93, 58, 127, 131, 61, 55, 107, 50, 394
5	13, 343, 451, 327, 308, 199, 296, 307
6	15, 19, 129, 254, 256, 312, 275, 19, 210, 26
7	130, 86, 113, 217, 259, 163, 308

Figure 2 gives the clustering diagram of some bicycle stations produced after clustering. In Figure 2, the horizontal coordinates represent the latitude of the bicycle stations, and the vertical coordinates represent the longitude of the bicycle stations. The different colors in Figure 2 represent clusters of different bicycle stations. Stations of the same color belong to one category. The middle part has more points of the same color. It indicates that the geospatial dataset used has a higher density of intermediate stations. The surrounding areas have fewer points of the same color. It indicates that the locations are more geographically isolated. There are fewer bicycle stations to cluster, so the nearby stations are clustered together to form a class. After getting the clustering information of the bike-sharing stations in the dataset based on the geographical location, the amount of bike-sharing rentals in the clustering information is used as input to TCN.

3.2. TCN-Based Prediction of Bike-Sharing Rentals. We predict the amount of bike-sharing rentals at each station. Historical data is used to predict future rental volumes. Classical algorithms, such as LSTM and RNN, are weak in capturing the past information of the data, resulting in inaccurate prediction results. A difficulty with traffic prediction models is that if the input sequence is too long, then the neural network loses previous traffic information. We use the dilation convolution method to solve this difficulty. By expanding the convolutional session, the perceptual field of view is increased to a larger size, for retaining more semantic information. In addition, we use causal convolution to solve the difficulty that the lengths of the input and output sequences are inconsistent.

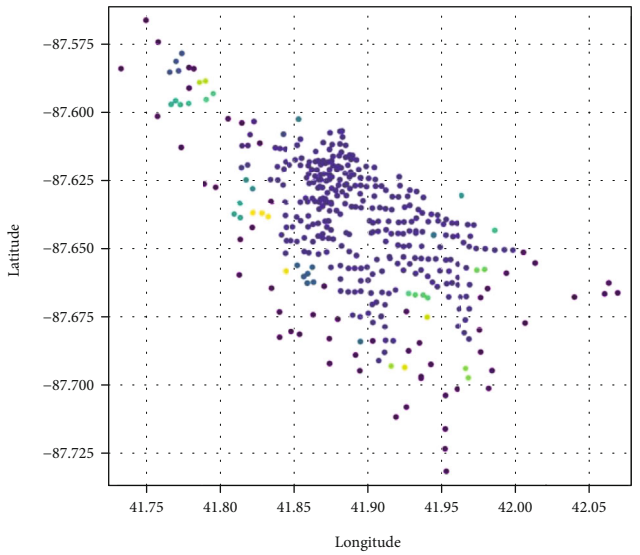

FIGURE 2: Clustering results of shared bike stations in the dataset according to the DBSCAN clustering algorithm.

In summary, the TCN used in the paper has 3 features. (1) The input sequence and the output sequence are kept consistent. In implementing full convolution, the input layer and the hidden layer are identical. In addition, 0 is used for padding to ensure that the subsequent layers are kept at the same length as the previous layers. (2) When predicting, the perceptual field of view is increased even more. To obtain a piece of longer history information, dilation convolution is introduced to construct a very deep neural network or a very large convolution kernel. (3) To prevent overfitting, the ReLu activation function and Dropout are employed. Therefore, the proposed structure consists of an input layer, causal convolution layer, dilation convolution layer, normalization layer, Dropout layer, ReLu layer, and output layer, as shown in Figure 3.

3.2.1. Input Layer of TCN. Based on the cluster information obtained by the DBSCAN algorithm clustering, the number of shared bicycle rentals belonging to all stations passing through a cluster is fed into the input layer of the TCN. The input layer is denoted **X**. Cluster 3 with 4 stations are used as an example. **X** is a two-dimensional matrix of size 4∗24. The number 4 represents the 4 stations. The number 24 represents a 24-hour day. The values in matrix **X** represent the number of rentals at each site in the dataset. The data in the input layer is shown in Equation (1).

$$X = \begin{bmatrix} x_{0,1} & x_{0,2} & x_{0,3} & x_{0,4} & \cdots & \cdots & x_{0,21} & x_{0,22} & x_{0,23} & x_{0,24} \\ x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & \cdots & \cdots & x_{1,21} & x_{1,22} & x_{1,23} & x_{1,24} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & \cdots & \cdots & x_{2,21} & x_{2,22} & x_{2,23} & x_{2,24} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} & & & x_{3,21} & x_{3,22} & x_{3,23} & x_{3,24} \end{bmatrix}. \tag{1}$$

In Equation (1), $[x_{0,1}\, x_{0,2}\, x_{0,3}\, x_{0,4} \cdots \cdots x_{0,21}\, x_{0,22}\, x_{0,23}\, x_{0,24}]$ is a row of the matrix **X**. The column subscripts of the row elements represent the 24 hours of the day. The row elements represent the number of rentals at a particular shared bike station from 0 : 00 to 24 : 00. Using cluster 3 as an example, the first row in Equation (1) represents the number of rentals per hour from 0 : 00 to 24 : 00 for the bike-share station with ID 260.

In Equation (1), $\begin{bmatrix} x_{0,1} & x_{1,1} & x_{2,1} & x_{3,1} \end{bmatrix}^T$ represents a column of the input matrix **X**. The 4 rows of this column correspond to the number of shared bicycle rentals from point 0 : 00 to 1 : 00 at each of the 4 stations. The first column in Equation (1) represents the number of rentals from 0 : 00 to 1 : 00 for the four stations in cluster 3, whose IDs are 260, 502, 290, and 123. The result of assigning matrix **X** to the Chicago Divvy bike-sharing dataset is shown in Equation (2).

$$X = \begin{bmatrix} 2 & 2 & 1 & 1 & \cdots & \cdots & 15 & 11 & 8 & 4 \\ 3 & 7 & 2 & 0 & \cdots & \cdots & 17 & 15 & 9 & 7 \\ 5 & 3 & 1 & 0 & \cdots & \cdots & 22 & 18 & 11 & 9 \\ 6 & 5 & 3 & 2 & \cdots & \cdots & 14 & 10 & 9 & 6 \end{bmatrix}. \tag{2}$$

3.2.2. Causal Convolution Layer of TCN. The first layer of TCN is a one-dimensional full convolution. The input matrix **X** is mapped through the full convolution network to the hidden layer. The dimensionality of the data does not change. Also to ensure the causality of the input and output, the TCN only performs convolutional operations on the input matrix **X** at the current moment as well as at previous moments.

Figure 4 shows the computational structure of the causal convolution. The input content of the causal convolution is shown in Equation (3). The computation process of the causal convolution is shown in Equations (4), (5), and (6).

$$X = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad \cdots \quad \cdots \quad x_{21} \quad x_{22} \quad x_{23} \quad x_{24}]. \tag{3}$$

In Equation (3), x_1 is the 1st column in the input matrix, i.e.,$x_1 = \begin{bmatrix} x_{0,1} & x_{1,1} & x_{2,1} & x_{3,1} \end{bmatrix}^T$.

$$j_1 \cdots j_t = F(x_1 \cdots x_t), \tag{4}$$

$$l_1 \cdots l_t = F(j_1 \cdots j_t), \tag{5}$$

$$y_1 \cdots y_t = F(l_1 \cdots l_t). \tag{6}$$

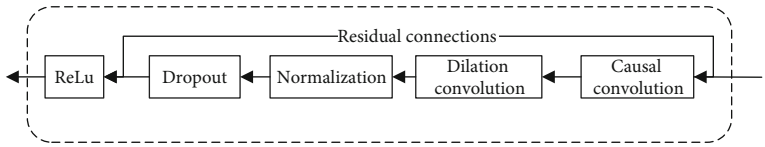

Figure 3: The basic structural unit of the proposed TCN.

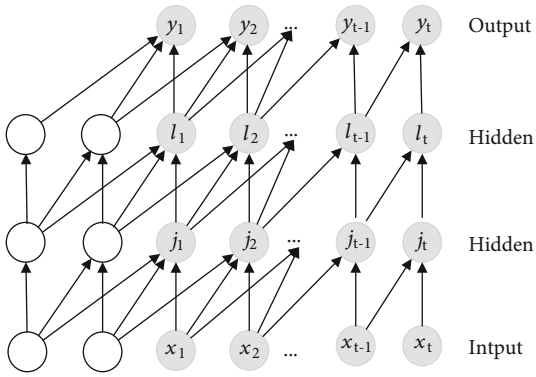

Figure 4: The structure of causal convolution layer of TCN.

As shown in Figure 4, j_t represents the result of the convolution when the input is x_t. l_t represents the result of the convolution when the input is j_t. y_t represents the result of the convolution when the input is l_t. They are all vectors. The dashed line represents the zero complement operation. The inputs and outputs are guaranteed to be the same size.

3.2.3. Dilation Convolution Layer of TCN. To keep the data input and output consistent, very deep network structures or very large convolutional kernels are required, when the amount of data is large. It places a significant burden on the computation of the model. Therefore, we introduce dilation convolution on top of causal convolution. The most important feature of the dilation convolution is that the perceptual field is expanded by controlling the dilation factor. In 1-dimensional dilation convolution, the dilation factor is equivalent to making the convolution kernel larger. The dilation factor is usually set to an exponential form of 2, e.g., 1, 2, 4, 8,…,2^i. When the dilation factor is equal to 1, the dilation convolution becomes a normal convolution operation. When the dilation factor becomes larger, the range of the field of perception also becomes larger. The output of the convolution will be linked to a long history of inputs.

When dealing with traffic data, the structure of the dilation convolution layer is shown in Figure 5. The first layer is the causal convolution layer. The output of the causal convolution layer is used as the input to the dilation convolution layer. As the number of layers of the dilation convolution increases, the length of the state and historical input data of the hidden layer increases exponentially. At the same time, the hidden layers will rapidly become smaller. The input data is represented as more high-dimensional.

Equation (7) is the formula for dilation convolution. Equation (8) represents the relationship between the number of layers of the network and the size of the convolution kernel. The convolution kernel size, the dilation factor, and the number of layers of the network are denoted as k, d, and i, respectively. An element of the input and the output of the causal convolution is denoted as s and y, respectively. N is positive integers.

$$z_1 \cdots z_t = \sum_{i=0}^{k-1} W(i) * y_{s-d\cdot i}, \tag{7}$$

$$d = 2^{i-1}(i = 1, 2, \cdots N). \tag{8}$$

As shown in Figure 5, $z_1 \cdots z_t$ is the output of the expanded convolution when the input is $y_1 \cdots y_t$, which are vectors.

3.2.4. Normalization Layer of TCN. In the TCN model, as the training continues to iterate and the network deepens, the distribution of the activation input values before the nonlinear transformation is done will gradually shift or update. The overall distribution will gradually move towards the upper and lower limits of the range of values of the activation function. If the model uses a Sigmoid activation function, after the model has been trained, the activation value will be updated to around 1. Because a feature is overweighted. Subsequently, no matter how much the value is increased and iterated over, the activation value will remain around 1. Too large a change will not occur. The above phenomenon can lead to an activation function that is insensitive to the features, affecting the effectiveness of the TCN.

In this paper, normalization is applied after each dilation of the causal convolution. The distribution of the input data is pulled back to the standard normal distribution. The inputs of each layer of the neural network are kept in the same distribution. It improves the training speed and convergence rate and avoids the gradient converging to 0. The normalization formula is shown in Equations (9), (10), (11), (12), and (13).

The input is denoted as $P = \{p_1, p_2, \cdots, p_n\}$. The linear activation value s_{p_i} for a given sample p_i is calculated as shown in Equation (9), and the weight matrix and bias parameters are set as W_p and b, respectively.

$$s_{p_i} = W_l p_i + b. \tag{9}$$

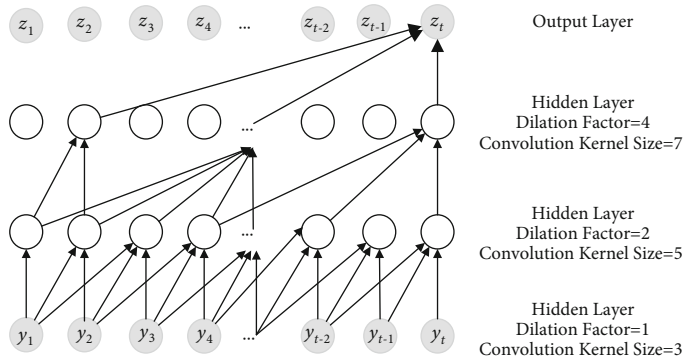

Figure 5: The structure of the dilation convolution layer of TCN.

According to Equations (10) and (11), the mean μ_p and variance ${\sigma_p}^2$ of the input data are calculated.

$$\mu_p = \frac{1}{u}\sum_{i=1}^{u} s_{p_i}, \tag{10}$$

$${\sigma_p}^2 = \frac{1}{u}\sum_{i=1}^{u}\left(s_{p_i} - \mu_p\right). \tag{11}$$

The activation value s_{pi} of each neuron within the hidden layer is transformed by a normal distribution. A new activation value $\hat{s}_{p_i}$ is obtained. In Equation (12), ε is a constant and is used to ensure the stability of the variance.

$$\hat{s}_{p_i} = \frac{s_{p_i} - \mu_p}{\sqrt{{\sigma_p}^2 + \varepsilon}}. \tag{12}$$

With Equations (9)–(12), the training speed of the model is boosted. The stability of the model is enhanced. The tuning process is optimized. However, this normal distribution transformation is equivalent to a linear function, which leads to a decrease in the expressiveness of the network. Therefore, to ensure that nonlinearity is obtained, two conditioning parameters γ and β are set in this paper, as shown in Equation (13). According to these two parameters, the activation values after the normal distribution transformation are then inverse transformed, for counteracting the side effects of the normal distribution transformation.

$$\hat{p}_i = \gamma\hat{s}_{p_i} + \beta. \tag{13}$$

3.2.5. Dropout Layer of TCN. Because the TCN uses causal convolution and dilation convolution, the number of network layers and intermediate layer results is increased. It not only results in an overall larger network structure and increased computational effort but also increases the memory used for training the network. To solve the above problem, the Dropout technique is used in this paper. As Equations (14) and (15) show, half of $z_1 \cdots z_t$ is assigned to 0. Dropout is randomly assigned to 0.

$$z_1 \cdots z_t = F(x_1 \cdots x_t), \tag{14}$$

$$z_2, z_4 \cdots z_{t-2}, z_t = 0. \tag{15}$$

3.2.6. ReLu Layer of TCN. ReLu is a commonly used activation function in neural networks. ReLu is adopted for the following reasons. (1) Traditional functions such as Sigmoid are used, which are complex to derive and computationally intensive when back-propagating the parameters to find the error gradient. When the ReLu activation function is used, the computational effort and computational time are saved. (2) In deeper network structures, gradient disappearance and gradient explosion can easily occur when back-propagating the parameters of the Sigmoid function, resulting in deeper network structures that are difficult to train. (3) The ReLu function being employed leads to some neurons having an output of 0. The network structure becomes sparse. Interdependencies between parameters are reduced. The problem of overfitting is alleviated. (4) The ReLu function is more easily optimized. Because the ReLu function is segmented linearly, it is easier to derive. Traditional Sigmoid functions tend to discard information in the propagation process.

After Dropout, as the ReLu function is used, any values less than 0 in $z_1, z_3 \cdots z_{t-3}, z_{t-1}$ become 0, and any values greater than 0 are the values themselves. Next, $z_1' \cdots z_t'$ is used as input to obtain $k_1 \cdots k_t$, and $k_1 \cdots k_t$ is treated in the same way as $z_1 \cdots z_t$ to obtain $m_1 \cdots m_t$. The above calculation process is shown in Equations (16), (17), (18), (19), and (20), where $x_1 \cdots x_t$ is the input, $z_1' \cdots z_t'$ and $k_1' \cdots k_t'$ are $z_1 \cdots z_t$ and $k_1 \cdots k_t$ after the ReLu activation function, and $m_1 \cdots m_t$ is the output.

$$z_1' \cdots z_t' = \mathrm{Relu}(z_1 \cdots z_t), \tag{16}$$

$$k_1 \cdots k_t = F\left(z_1' \cdots z_t'\right), \tag{17}$$

$$k_2, k_4 \cdots k_{t-2}, k_t = 0, \tag{18}$$

$$k_1' \cdots k_t' = \mathrm{Relu}(k_1 \cdots k_t), \tag{19}$$

$$m_1 \cdots m_t = F\left(k_1' \cdots k_t'\right). \tag{20}$$

3.2.7. Output Layer of TCN. After the ReLu activation function, the result of the ReLu function, m_t, is output by the output layer. The operation of the ReLu activation function layer and the final output layer is shown in Equation (21). The model finally obtains the prediction result s_t

$$m_1 \cdots m_t = s_1 \cdots s_t, \tag{21}$$

where $m_1 \cdots m_t$ is the result of the ReLu calculation and $s_1 \cdots s_t$ is the result of the output layer.

4. Numerical Evaluation and Discussion

4.1. Experimental Data and Preprocessing. The Chicago Divvy bike-sharing dataset is used in this paper. Divvy is part of the official bike-sharing system of the Chicago Department of Transportation (CDOT) and operated by Motivate, a leading bike-sharing manufacturer. There are currently more than 6,000 bikes available at over 580 stations. The bikes are available at fixed stations, i.e., docked bikes. The process of using the bicycle consists of obtaining a permit, unlocking at a station, riding, and returning at any station. The data for each year includes both station and trip data. Each station contains five fields, which are name, station name; latitude, station latitude; longitude, station longitude; dpcapacity, number of total docks at each station; and online date, date the station went live in the system. Each trip contains twelve fields, which are trip_id, ID attached to each trip taken; starttime, day and time trip started, in CST; stoptime, day and time trip ended, in CST; bikeid, ID attached to each bike; tripduration, time of trip in seconds; from_station_name, name of station where trip originated; to_station_name, name of station where trip terminated; from_station_id, ID of station where trip originated; to_station_id, ID of station where trip terminated; usertype, "Customer" is a rider who purchased a 24-Hour Pass, "Subscriber" is a rider who purchased an Annual Membership; gender, gender of rider; and birthyear, birth year of rider.

First, the geographic location information in this dataset was clustered, and then, bike-share rentals were predicted for different stations in the same clustered cluster. The data in the dataset between 1 January 2016 and 31 December 2016 were selected. The 24 clusters in the clustering result each contained sites ranging from 2 to 20. Clusters with 7-17 sites generally predominated. Cluster 7, which contains 7 sites, was selected as an example. Cluster 7 contains sites with IDs of 130, 86, 113, 217, 259, 163, and 308 as shown in Table 2. The number of bicycles rented from the seven stations with IDs 130, 86, 113, 217, 259, 163, and 308 during 10 : 00-11 : 00 on December 1, 2016, was 8, 1, 15, 1, 6, 17, and 8, respectively. The time interval was set to 1 hour, and the number of bicycles rented from cluster 7 at each period was calculated in this way.

Bicycle rental volume data, from January 1 to December 20, 2016, was selected for training the DBSCAN-TCN model. When the model was trained, information on the number of shared bicycle rentals at each station was fed into the DBSCAN-TCN model for prediction. The predicted results are compared with the true values. The error rate and root mean squared logarithmic error of the predicted and true values are calculated to evaluate the performance of the proposed prediction model.

Table 2: Number of bikes rented out in cluster 7 on 1 Dec. 2016.

Time	Station ID						
	130	86	113	217	259	163	308
6 : 00-7 : 00	1	4	0	0	1	0	1
7 : 00-8 : 00	5	9	0	3	1	1	2
8 : 00-9 : 00	3	13	3	3	1	2	3
9 : 00-10 : 00	12	28	4	3	2	2	1
10: 00-11: 00	8	1	15	1	6	17	8
11: 00-12: 00	13	6	5	1	11	7	19
12: 00-13: 00	6	8	10	13	16	7	2
13 : 00-14 : 00	9	11	13	7	2	4	3
14 : 00-15 : 00	10	7	12	13	16	6	8

Table 3: Parameter setting for the DBSCAN-TCN model.

Parameter	Setting
Activation functions	ReLu
Number of time steps	time_step = 24
Number of hidden cells per layer	24
Input layer dimension	input_size = 2
Output layer latitude	output_size = 1
Number of cycles	max_epoch = 100
Number of layers of the temporal convolutional network	4 layers
Convolutional kernel size	3
Dropout	0.5
Number of samples per batch	bitch_size = 24
Learning rate	learning_rate = 1

4.2. Implementation of DBSCAN-TCN Prediction Model with Pytorch. Pytorch has undergone rapid development in recent years and is used in a wide range of applications [42] Pytorch has many advantages. The Pytorch library is relatively easy to understand and works seamlessly with NumPy and SciPy. The tensor-based GPU acceleration is very powerful. During the calculation of network gradients, the network structure can be designed dynamically, without the need to build static graphs from scratch. Based on the simple and flexible design, the GPU can be used to accelerate the training of the network. Therefore, the proposed DBSCAN-TCN model is implemented using the Pytorch deep learning framework.

We use Pytorch to build the DBSCAN-TCN model and use the model to predict the amount of bike-sharing rentals. The implementation process involves setting up a set of

Table 4: Specific implementation steps of the DBSCAN-TCN model.

Step	Content
Step 1	The bike-sharing data is collated into a suitable input format. The data is fed into the model according to chronological order.
Step 2	The structure and parameters of the model are determined. For example, the dilation factor, the number of network layers, the size of the convolutional kernel, the number of layers in the hidden layers, and the number of neurons per layer.
Step 3	The learning rate, the appropriate optimization technique (Dropout), and the appropriate activation function (ReLu) are all selected.
Step 4	The training dataset is used to train the optimization model. The parameters of the prediction model are trained.
Step 5	The validation dataset is used to verify the model predictions. If the prediction is good, the model goes to Step 6; otherwise, it returns to Step 2.
Step 6	The model with the best prediction and the test dataset are used to predict the number of shared bicycle rentals.

Table 5: Performance of the proposed model on cluster 4 when choosing different Dropout rates.

Dropout	RMLSE	ER
0.4	0.268	0.253
0.5	0.241	0.231
0.6	0.312	0.301

TCN network structural units. The size of the convolutional kernel is set to 3. The number of layers of the causal and dilation convolutional networks is set to 4. The number of nodes in the hidden unit is set to 24. The input and output tensor dimensions of each TCN unit are set to 2 and 1, respectively. The activation function is set to ReLu. The number of cycles was set to 100. The number of time steps and the number of training samples per batch were both set to 24. As shown in Table 3, the learning rate was set to 1, and the Dropout was chosen to be 0.5.

Table 4 gives the six steps for training and using the DBSCAN-TCN model for predicting the amount of bike-sharing rentals.

4.3. Experimental Evaluation Criteria. The evaluation metrics used in this paper are RMLSE (root mean squared logarithmic error) and ER (error rate), which are shown in Equations (22) and (23), XCi,t are the true values of the number of bike-share rentals in a cluster Ci over time t, and $X^{\wedge}_{C_{i,t}}$ are the predicted values generated by the model. The predicted length of time is denoted as m, $m = 240$, and the time horizon of a day is denoted as T, $T = 24$. The number of clusters is denoted as i, e.g., C_1 denotes the first cluster.

$$\mathrm{RMLSE} = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\log\left(X^{\wedge}_{C_{i,t}}+1\right)-\log\left(X_{C_{i,t}}+1\right)\right)^{2}}, \tag{22}$$

$$\mathrm{ER} = \frac{1}{T}\sum_{1}^{T}\frac{\sum_{i=1}^{m}\left|X^{\wedge}_{C_{i,t}}-X_{C_{i,t}}\right|}{\sum_{i=1}^{m}X_{C_{i,t}}}. \tag{23}$$

4.4. Effect of Hyperparameters on Experimental Results. The function of Dropout is to prevent all the feature extractors from acting together and causing some features to be scaled up or down all the time. Dropout not only prevents these problems from occurring during training but also increases the ability of the model to generalize. For the selection of Dropout rate, cluster 4 was chosen in this paper to validate as a representative. The reason is that cluster 4 contains the most number of bicycle stations, with 20 stations. During the training of the model, cluster 4 would be very computationally intensive. Dropout is therefore most evident for cluster 4. Table 5 shows that with a Dropout rate equal to 0.5, the trained model works best for cluster 4 and has the lowest error.

4.5. Experimental Results and Analysis. When people travel, they do not always use a shared bike to reach their destination directly. Usually, people change to other means of transport to reach their destination. For work and school, it is more common. Therefore, the use of shared bikes is inseparable from the period. During work and school hours, the number of shared bicycle rentals increases substantially. However, at other times of the day, bicycle-sharing rentals gradually return to normal levels. It is just the pattern of bicycle rental during normal working hours. When it is a day off, people are usually at home and resting. Most people are off work, although overtime can occur on weekends. Figure 6 shows the average number of bike-sharing rentals across the city at different times of the day in December 2016.

As can be seen in Figure 6, the curve changes are broadly the same throughout the weekdays in terms of bike-share rentals. Each inflection point is also roughly the same. On weekends, the number of shared bicycle rentals is much lower than on weekdays. It means that people are resting and not going out. The shared bikes are not being used. The curves for Saturday and Sunday are also broadly similar and have roughly the same inflection points.

Figure 6 shows that the number of shared bicycle rentals is lower in the early hours of a day. After 7 AM, the number of shared bicycle rentals gradually increases. By the end of the day, the magnitude of the change in rentals becomes even greater. After work hours, the number of bicycle-sharing rentals steadily drops back to normal levels. At the end of the work, the number of shared bicycle rentals rises again. By the end of the day, there will be very little use of shared bikes, because people are resting.

As can be seen in Figure 7, the difference between the predicted and true value curves for the station with ID 47

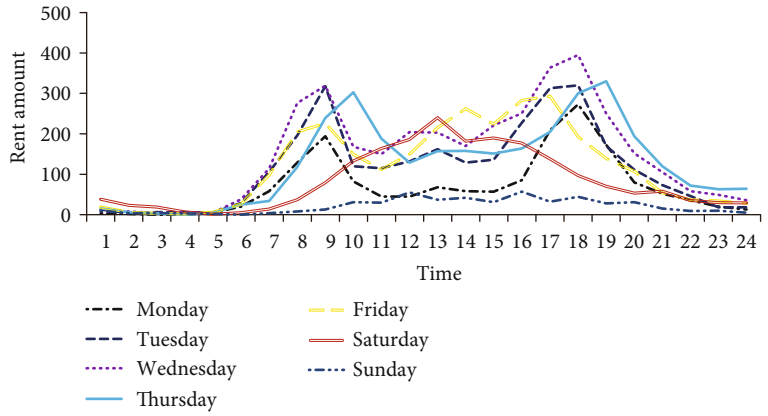

Figure 6: The rent amount of shared bikes varies over time, during different days of a week.

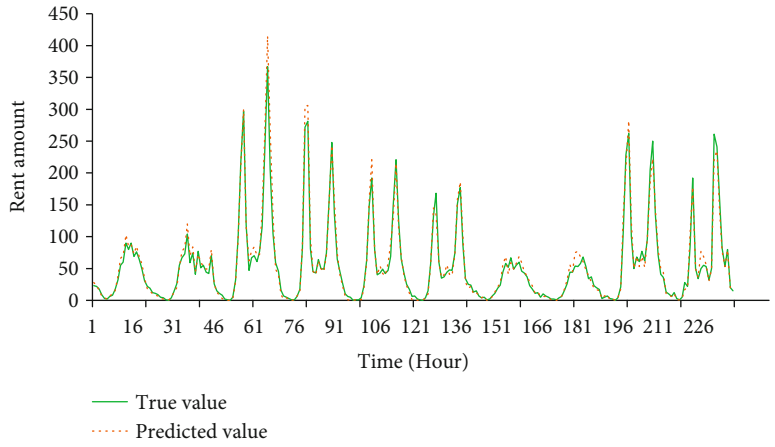

Figure 7: Predicted and true values of shared bicycle rentals at the site with ID 47 in cluster 2.

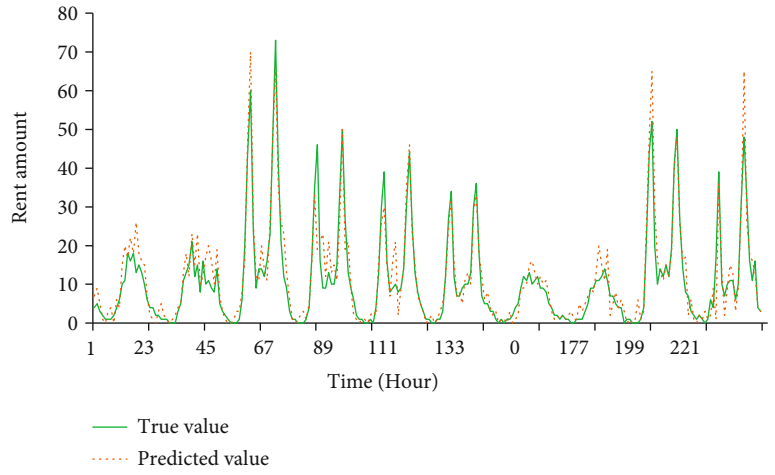

Figure 8: Predicted and true values of shared bicycle rentals at the site with ID 130 in cluster 7.

in cluster 2 is very small. The trends of the predicted and true values are also very similar. The range of variation between the predicted and true values is also very similar. It indicates that the proposed DBSCAN-TCN model has a good prediction effect. The horizontal and vertical coordinates in Figure 7 represent time and rental volume, respectively. The dashed line represents the predicted value, and the solid line represents the true value.

As shown in Figure 8, the prediction results for the site with ID 130 in cluster 7 are worse compared to the prediction for the site with ID 47 in cluster 2. In Figure 8, the difference between the predicted curve and the curve of the true

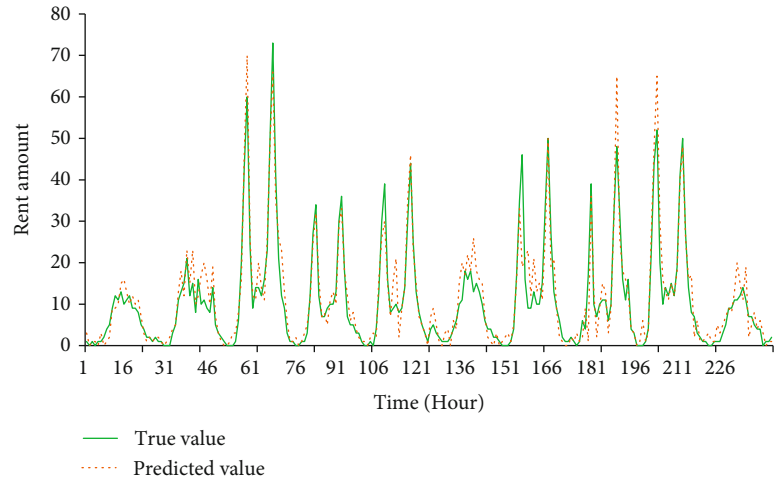

Figure 9: Predicted and true values of shared bicycle rentals at the site with ID 13 in cluster 5.

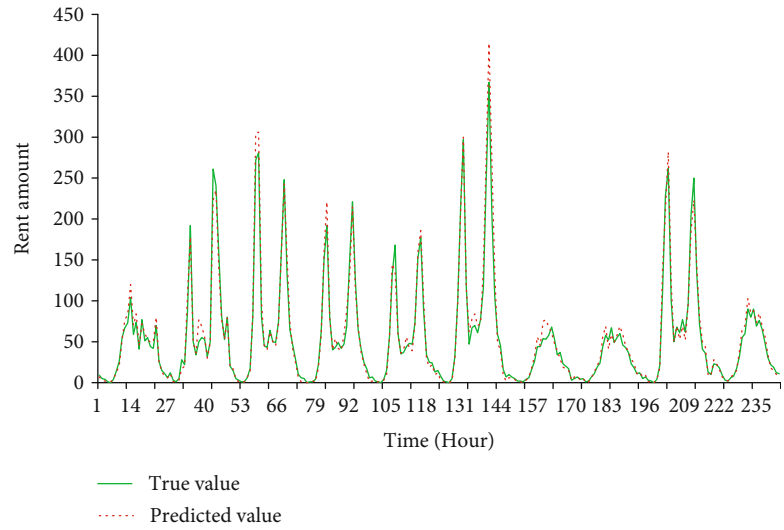

Figure 10: Predicted and true values of shared bicycle rentals at the site with ID 15 in cluster 6.

Table 6: Effect of different convolution kernel sizes on performance.

Convolutional kernel size	3		4		5	
	RMLSE	ER	RMLSE	ER	RMLSE	ER
Cluster 1	0.278	0.268	0.268	0.253	0.288	0.278
Cluster 2	0.242	0.255	0.232	0.215	0.252	0.265
Cluster 3	0.232	0.241	0.212	0.211	0.248	0.256
Cluster 4	0.249	0.238	0.241	0.231	0.254	0.265
Cluster 5	0.266	0.267	0.256	0.241	0.272	0.271
Cluster 6	0.247	0.268	0.234	0.256	0.251	0.273
Cluster 7	0.226	0.235	0.206	0.229	0.235	0.248

value is large. The reason for this occurrence is due to the small number of sites within cluster 7. It results in an inadequate capture of the information in the data.

Figures 9 and 10 show the predicted and true values for the number of shared bike stations rented for ID 13 in cluster 5 and ID 15 in cluster 6. The horizontal coordinates represent time, and the vertical coordinates represent the number of shared bicycle rentals at the stations in the cluster with the corresponding IDs.

The size of the convolution kernel has an impact on the performance of the model. The average performance evaluation of the model for each cluster is given in Table 6, for convolutional kernel sizes of 3, 4, and 5, respectively. It can be seen from Table 6 that as the convolutional kernel size increases, both the RMLSE and the ER increase. The average RMLSE and ER are both the largest for the clusters containing a small number of stations as they increase. It indicates that the effect of a small number of stations on the prediction of shared bicycle rentals is significant. The average RMLSE and ER of shared bicycle rentals for clusters with a convolution kernel size of 3 are the lowest. The small RMLSE and ER indicate that the model is predicting well.

In summary, taking the seven clusters in the dataset as an example, we use a Kalman filter model [43], an ARMA

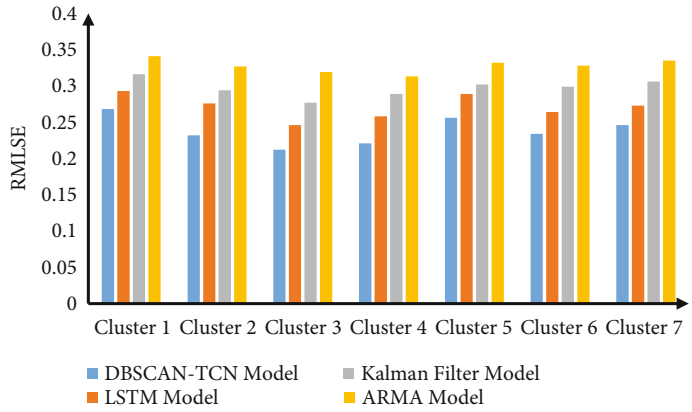

Figure 11: Comparison of the RMLSE of four forecasting models.

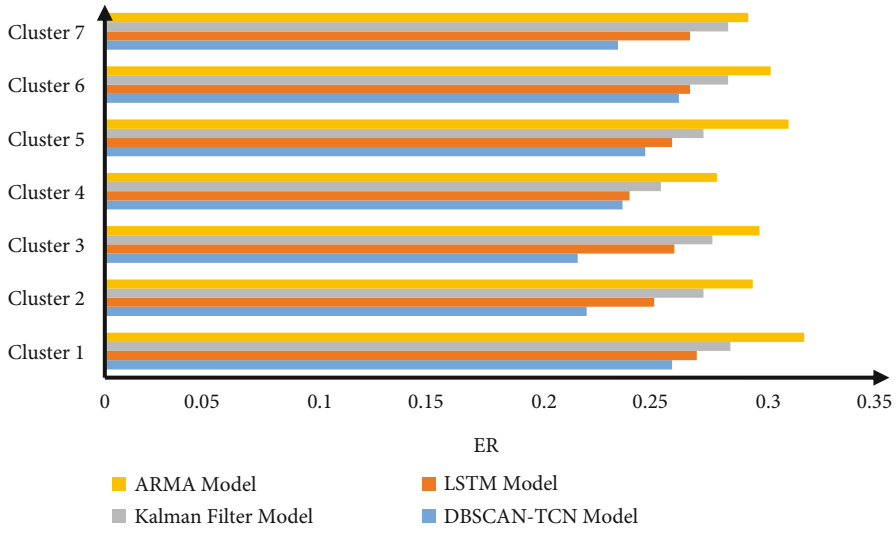

Figure 12: Comparison of the ER of four forecasting models.

model [44], and an LSTM model [45] to predict the average number of shared bicycle rentals for all stations within each cluster. Both ER and RMLSE were accounted for. The comparison results compared with the proposed DBSCAN-TCN model are shown in Figures 11 and 12.

Between Figures 11 and 12, it can be seen that the RMLSE and ER values of the DBSCAN-TCN model are lower than the Kalman filter model, the ARMA model, and the LSTM model. It indicates that the DBSCAN-TCN model is better than the Kalman filter model, the ARMA model, and the LSTM model in predicting the number of shared bicycle rentals, and the RMLSE and ER of the LSTM model are lower than the Kalman filter model and the ARMA model. The ARMA model has the worst prediction results. Therefore, it can be found that the proposed DBSCAN-TCN model in this paper has the best prediction effect.

5. Conclusions

In this paper, a DBSCAN-TCN model is proposed to forecast the amount of bike-sharing rentals. The proposed model uses the DBSCAN clustering algorithm to cluster the geographical data of bike-sharing stations. Based on the DBSCAN clustering results of the shared bicycle sites, the shared bicycle rentals of different sites belonging to the same cluster are transformed into the input data of the TCN. Using the trained DBSCAN-TCN model, the bike-sharing rentals are predicted for each station within each cluster. The effect of convolution kernel size on the model was also verified through the experiments. The experimental results show that the model predicts best when the convolution kernel size is 3. The performance of the model is evaluated by calculating and comparing the average RMLSE and ER of the sites in seven different clusters. The proposed DBSCAN-TCN model is compared with the Kalman filter model, the ARMA model, and the LSTM prediction model. The proposed model outperforms the classical three forecasting models in terms of both RMLSE and ER.

The main research of the paper is the prediction of the rental volume of the bike-sharing system. By analyzing the Chicago Divvy bike-sharing dataset, the DBSCAN-TCN model is proposed for the prediction of rental volumes. Certain research results were achieved. However, since a bicycle sharing system is a complex system, various factors need to

be considered. The study of bicycle-sharing rental volume still needs further improvement. In subsequent studies, other factors such as weather, temperature, and social holidays will be considered. A hybrid model based on the urban public bicycle prediction model is a good alternative. In the future, the model will have three main components to be considered in the research process. The first part will be a two-factor clustering of stations based on their geographical location and a historical transition model. The second part will predict the overall city bicycle hire by taking into account the time of day, weather, temperature, and wind speed factors. The third part seeks to find an inference model based on a coefficient of variation function. The model will be used to find the proportion of overall rental volume in each cluster and thus to predict the amount of bicycle rental in each cluster.

Data Availability

Previously reported Chicago Divvy bike-sharing dataset data were used to support this study and are available at https://www.divvybikes.com. These prior studies and datasets are cited at relevant places within the text as references [29].

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The project is supported by the Science and Technology Funds from Liaoning Education Department (No. LJ2020033 and No. LJKZ0449) and the National Social Science Foundation (No. 19CKS050).

References

[1] C. S. Lai, Y. Jia, Z. Dong et al., "A review of technical standards for smart cities," *Clean Technologies*, vol. 2, no. 3, pp. 290–310, 2020.

[2] P. Wang, C. Lin, M. S. Obaidat, Z. Yu, and Q. Zhang, "Contact tracing incentive for COVID-19 and other pandemic diseases from a crowdsourcing perspective," *IEEE Internet of Things Journal*, vol. 8, no. 21, 2021.

[3] C. Lin, G. Han, J. Du, T. Xu, and Z. Lv, "Spatiotemporal congestion-aware path planning toward intelligent transportation systems in software-defined smart city IoT," *IEEE Internet of Things Journal*, vol. 7, 2020.

[4] J. C. F. de Guimarães, E. A. Severo, L. A. Felix Júnior, W. P. L. B. da Costa, and F. T. Salmoria, "Governance and quality of life in smart cities: towards sustainable development goals," *Journal of Cleaner Production*, vol. 253, p. 119926, 2020.

[5] L. Zhao, H. Li, N. Lin, M. Lin, C. Fan, and J. Shi, "Intelligent content caching strategy in autonomous driving toward 6G," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2021.

[6] L. Zhao, W. Zhao, A. Hawbani et al., "Novel online sequential learning-based adaptive routing for edge software-defined vehicular networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 5, pp. 2991–3004, 2021.

[7] C. K. Toh, J. A. Sanguesa, J. C. Cano, and F. J. Martinez, "Advances in smart roads for future smart cities," *Proceedings of the Royal Society A*, vol. 476, no. 2233, 2020.

[8] M. A. Ahad, S. Paiva, G. Tripathi, and N. Feroz, "Enabling technologies and sustainable smart cities," *Sustainable Cities and Society*, vol. 61, 2020.

[9] Z. Ullah, F. Al-Turjman, L. Mostarda, and R. Gagliardi, "Applications of artificial intelligence and machine learning in smart cities," *Computer Communications*, vol. 154, pp. 313–323, 2020.

[10] D. Zou and D. Gong, "Differential evolution based on migrating variables for the combined heat and power dynamic economic dispatch," *Energy*, vol. 238, 2022.

[11] R. Yu, D. Ye, Z. Wang et al., "CFFNN: cross feature fusion neural network for collaborative filtering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 99, 2021.

[12] C. Badii, P. Bellini, A. Difino, and P. Nesi, "Sii-Mobility: an IoT/IoE architecture to enhance smart city mobility and transportation services," *Sensors*, vol. 19, no. 1, 2019.

[13] M. A. Jan, M. Zakarya, M. Khan et al., "An AI-enabled lightweight data fusion and load optimization approach for Internet of Things," *Future Generation Computer Systems*, vol. 122, pp. 40–51, 2021.

[14] M. Merenda, C. Porcaro, and D. Iero, "Edge machine learning for AI-enabled IoT devices: a review," *Sensors*, vol. 20, no. 9, 2020.

[15] H. Nguyen-An, T. Silverston, T. Yamazaki, and T. Miyoshi, "IoT traffic: modeling and measurement experiments," *IoT*, vol. 2, no. 1, pp. 140–162, 2021.

[16] S. U. Jamil, M. A. Khan, and S. U. Rehman, "Intelligent task off-loading and resource allocation for 6G smart city environment," *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, 2020, pp. 441–444, Sydney, NSW, Australia, 2020.

[17] B. Charyyev and M. H. Gunes, "IoT traffic flow identification using locality sensitive hashes," *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6, Dublin, Ireland, 2020.

[18] Z. Zou, Y. Jin, P. Nevalainen, Y. Huan, J. Heikkonen, and T. Westerlund, "Edge and fog computing enabled AI for IoT-an overview," *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2019, pp. 51–56, Hsinchu, Taiwan, 2019.

[19] D. Zou, S. Li, X. Kong, H. Ouyang, and Z. Li, "Solving the dynamic economic dispatch by a memory-based global differential evolution and a repair technique of constraint handling," *Energy*, vol. 147, pp. 59–80, 2018.

[20] A. Ladha, P. Bhattacharya, N. Chaubey, and U. Bodkhe, "*IIGPTS*: IoT-based framework for intelligent green public transportation system," *Proceedings of First International Conference on Computing, Communications, and Cyber-Security (IC4S 2019)*, , pp. 183–195, Springer, 2020.

[21] C. Lin, G. Han, X. Qi, M. Guizani, and L. Shu, "A distributed mobile fog computing scheme for mobile delay-sensitive applications in SDN-enabled vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5481–5493, 2020.

[22] V. Puri, S. S. Jagdev, J. G. Tromp, and C. Van Le, "Smart Bicycle: IoT-Based Transportation Service," in *Intelligent Computing in Engineering*, pp. 1037–1043, Springer, 2020.

[23] H. Tahaei, F. Afifi, A. Asemi, F. Zaki, and N. B. Anuar, "The rise of traffic classification in IoT networks: a survey," *Journal of Network and Computer Applications*, vol. 154, p. 102538, 2020.

[24] R. J. Bonilla-Alicea, B. C. Watson, Z. Shen, L. Tamayo, and C. Telenko, "Life cycle assessment to quantify the impact of technology improvements in bike-sharing systems," *Journal of Industrial Ecology*, vol. 24, no. 1, pp. 138–148, 2020.

[25] O. O'Brien, J. Cheshire, and M. Batty, "Mining bicycle sharing data for generating insights into sustainable transport systems," *Journal of Transport Geography*, vol. 34, pp. 262–273, 2014.

[26] A. A. Kadri, I. Kacem, and K. Labadi, "A branch-anDBSCAN-bound algorithm for solving the static rebalancing problem in bicycle-sharing systems," *Computers & Industrial Engineering*, vol. 95, pp. 41–52, 2016.

[27] G. Erdoğan, M. Battarra, and R. Wolfler Calvo, "An exact algorithm for the static rebalancing problem arising in bicycle sharing systems," *European Journal of Operational Research*, vol. 245, no. 3, pp. 667–679, 2015.

[28] L. Li and M. Shan, "Bidirectional incentive model for bicycle redistribution of a bicycle sharing system during rush hour," *Sustainability*, vol. 8, no. 12, p. 1299, 2016.

[29] D. Zhang, C. Yu, J. Desai, H. Y. Lau, and S. Srivathsan, "A time-space network flow approach to dynamic repositioning in bicycle sharing systems," *Transportation Research Part B:Methodological*, vol. 103, pp. 188–207, 2017.

[30] A. Faghih-Imani and N. Eluru, "Analysing bicycle-sharing system user destination choice preferences: Chicago's Divvy system," *Journal of Transport Geography*, vol. 44, pp. 53–64, 2015.

[31] J. Zhang, X. Pan, M. Li, and P. S. Yu, "Bicycle-sharing system analysis and trip prediction," in *2016 17th IEEE International Conference on Mobile Data Management (MDM)*, pp. 174–179, Porto, Portugal, 2016.

[32] J. Froehlich, J. Neumann, and N. Oliver, "Sensing and predicting the pulse of the city through shared bicycling," in *IJCAI 2009, Proceedings of the, international joint conference on artificial intelligence*, pp. 1420–1426, Pasadena, California, USA, 2009.

[33] A. Kaltenbrunner, R. Meza, J. Grivolla, J. Codina, and R. Banchs, "Urban cycles and mobility patterns: exploring and predicting trends in a bicycle-based public transport system," *Pervasive and Mobile Computing*, vol. 6, no. 4, pp. 455–466, 2010.

[34] P. Vogel, T. Greiser, and D. C. Mattfeld, "Understanding bike-sharing systems using data mining: exploring activity patterns," *Procedia - Social and Behavioral Sciences*, vol. 20, no. 6, pp. 514–523, 2011.

[35] J. W. Yoon, F. Pinelli, and F. Calabrese, "Cityride: a predictive bike sharing journey advisor," in *2012 IEEE 13th International Conference on Mobile Data Management*, pp. 306–311, Bengaluru, India, 2012.

[36] R. B. Noland, M. J. Smart, and Z. Guo, "Bikeshare trip generation in New York City," *Transportation Research Part A*, vol. 94, pp. 164–181, 2016.

[37] K. Gebhart and R. B. Noland, "The impact of weather conditions on bikeshare trips in Washington, DC," *Transportation*, vol. 41, no. 6, pp. 1205–1225, 2014.

[38] P. Borgnat, C. Robardet, P. Abry, P. Flandrin, J. B. Rouquier, and N. Tremblay, "A dynamical network view of Lyon's Vélo'v shared bicycle system," *Dynamics On and Of Complex Networks*, vol. 2, pp. 267–284, 2013.

[39] A. A. Campbell, C. R. Cherry, M. S. Ryerson, and X. Yang, "Factors influencing the choice of shared bicycles and shared electric bikes in Beijing," *Transportation Research Part C*, vol. 67, no. 6, pp. 399–414, 2016.

[40] M. Kaspi, T. Raviv, and M. Tzur, "Detection of unusable bicycles in bike-sharing systems," *Omega*, vol. 65, no. 12, pp. 10–16, 2016.

[41] G. Zhang, H. Yang, S. Li, Y. Wen, and F. Liu, "What is the best catchment area of bike share station? A study based on Divvy system in Chicago, USA," *2019 5th International Conference on Transportation Information and Safety (ICTIS)*, 2019, https://ieeexplore.ieee.org/document/8883774.

[42] N. Ketkar, "Introduction to Pytorch," in *Deep Learning with Python*, pp. 195–208, Springer, 2017.

[43] S. Mandal, P. Chandramohan, P. H. Yen, C. D. Jan, Y. P. Lee, and H. F. Lee, "Discussion and closure: application of Kalman filter to short-term tide level prediction," *Journal of Waterway, Port, Coastal, and Ocean Engineering*, vol. 124, no. 4, pp. 213-214, 1998.

[44] I. Rojas, O. Valenzuela, F. Rojas et al., "Soft-computing techniques and ARMA model for time series prediction," *Neurocomputing*, vol. 71, no. 4-6, pp. 519–537, 2008.

[45] B. S. Pramod and P. M. Mallikarjuna Shastry, "Stock price prediction using LSTM," *Test Engineering and Management*, vol. 83, pp. 5246–5251, 2021.