Automatic Traffic State Recognition Based on Video Features Extracted by an Autoencoder

Xiaoyu Cai, Qiongli Jing, Bo Peng, Yuanyuan Zhang, Yuting Wang, and Ju Tang

College of Traffic and Transportation, Chongqing Jiaotong University, Chongqing 400074, China

Correspondence should be addressed to Bo Peng; pengbo351@126.com

Received 29 July 2021; Revised 13 December 2021; Accepted 1 February 2022; Published 28 March 2022

Abstract

Video surveillance has become an important measure of urban traffic monitoring and control. However, due to the complex and diverse video scenes, traffic data extraction from original videos is a sophisticated and difficult task, and corresponding algorithms are of high complexity and calculation cost. To reduce the algorithm complexity and subsequent computation cost, this study proposed an autoencoder model which effectively reduces the video dimension by optimizing structural parameters; thus several traffic recognition models can conduct image processing work based on dimension-reduced videos. Firstly, an optimal autoencoder model with five hidden layers was constructed. Then, it was combined with a linear classifier, support vector machine, deep neural network, DNN linear classification method, and the k-means clustering method; thus, five traffic state recognition models were constructed: $A^*$-Linear, $A^*$-SVM, $A^*$-DNN, $A^*$-DNN_Linear, and $A^*$-k-means. Train and test results show that the accuracy rate and recall rate of $A^*$-linear, $A^*$-SVM, $A^*$-DNN, and $A^*$-DNN_Linear were 94.5%–97.1%, and the $F_1$ score was 94.4%–97.1%; besides, the accuracy rate, recall rate, and $F_1$ score of $A^*$-k-means were all approximately 95%, which suggests that the combination of the autoencoder $A^*$ and common classification or clustering methods achieve good recognition performance. Comparison was also implemented among the five models proposed above and four CNN-based models such as AlexNet, LeNet, GoogLeNet, and VGG16, which shows that the five proposed modes achieve $F_1$ scores of 94.4%–97.1%, while the four CNN-based models achieve $F_1$ scores of 16.7%–94%, indicating that the proposed light weight design methods outperform more complex CNN-based models in traffic state recognition.

1. Introduction

With the rapid development of urbanization and the sustainable development of a national economy, traffic congestion has increasingly become a key issue in large and medium-sized cities. To effectively control traffic operations and alleviate traffic congestion, it is very important to achieve traffic state recognition in a timely and effective manner. Currently, extensive studies have been conducted on traffic state recognition using a variety of methods, such as fuzzy logic in [1], the fuzzy C-means clustering algorithm in [2, 3], machine learning in [4, 5], and the K-means clustering algorithm in [6]. Data used in these studies were mainly from fixed surveillance systems such as cameras and vehicle detectors, as well as vehicle GPS in [7, 8], etc. With the enrichment of urban traffic surveillance video resources and the progress of image processing technology, traffic state recognition based on video images has gained wide attention. A large number of studies have been conducted on traffic state recognition based on video images. Traditionally, traffic state recognition was achieved by establishing certain algorithms to extract the traffic image characteristics in [9, 10]. Since these algorithms had high complexity, they were not able to cope with the rapid increase in mass traffic surveillance videos. To avoid the extraction, analysis, and modeling of complex image features, deep learning methods have recently been used to achieve rapid and effective image pattern recognition using large-scale image training. The convolutional neural network (CNN) is one of the most common deep learning methods and has achieved great success in image classification and object detection tasks in [11, 12]. However, to the best of our knowledge, a few studies...
have used the CNN in traffic state recognition based on video images.

Video images often have high dimensionality and contain complex and redundant information. To recognize the traffic states timely and effectively, attempts have been conducted to reduce the dimensionality of video images, thereby performing image pattern recognition based on the dimension-reduced data in [13]. Encoders, which are nonlinear and unsupervised neural network models that include an input layer, a hidden layer, and an output layer, can effectively reduce the dimensionality of video image data and realize image classification in [14].

Traffic data extraction from original videos is a sophisticated and difficult task because of the complex and diverse video scenes, and corresponding algorithms are of high complexity and calculation cost. In this study, to reduce the algorithm complexity and subsequent computation cost, an autoencoder model was proposed which effectively reduces the video dimension by optimizing structural parameters. Then, a traffic state recognition method was established by integrating the autoencoder with common classification and clustering methods. The train and test result shows that the proposed method has the advantages of a lightweight model structure and low calculation cost, which outperforms CNN models such as AlexNet and GoogLeNet. Therefore, it is suitable for traffic state recognition from videos.

2. Background

This section explains the relevant background of the video-based traffic state recognition, and related work are reviewed, including traditional traffic video processing, convolutional neural networks, and autoencoders.

The research results of video-based traffic state recognition are very fruitful. Among them, the earlier research mainly achieved traffic state recognition by establishing certain algorithms to extract the traffic image characteristics. By extracting a discrete cosine transform and its movement characteristics based on traffic surveillance videos, the hidden Markov model of fuzzy C-means was used to learn and recognize the traffic congestion state. A fuzzy logic method for traffic state recognition based on camera images was proposed. A traffic state recognition method was proposed, using traffic monitoring videos based on the fuzzy C-means clustering algorithm. A method recognizing three traffic states (free traffic flow, steady traffic flow, and forced traffic flow) based on images and video data was achieved in [15], and an image-based traffic density estimation method was proposed, in which a support vector machine classifier was used to classify traffic states into heavy, medium, and light densities. In general, this type of algorithms has high complexity and poor real-time video processing ability.

The convolutional neural network (CNN) has achieved rapid and effective image pattern recognition using large-scale image training, which can avoid the extraction, analysis, and modeling of complex image features. It is one of the most common deep learning methods and has achieved great success in image classification and object detection tasks. In 2012, Krizhevsky et al. brought out AlexNet, which ranked the first in the ImageNet image classification competition in [16]. Since then, the network structure has been improved, and other CNN models, such as visual geometry group (VGG) and GoogLeNet, have been proposed. A UAV image vehicle recognition method based on CNN and SVM was proposed in [17]. A satellite image vehicle detection method based on CNN and hard case mining was put forward in [18]. The authors applied MIT and Caltech vehicle datasets, and proposed a vehicle detection method in traffic scenes based on improved Faster R-CNN in [19]. The CNN has gained extensive interest in large-scale image classification, but less study has used the CNN in traffic state recognition based on video images.

Encoders, which are nonlinear and unsupervised neural network models, can effectively reduce the dimensionality of video image data in [20, 21]. A supervised autoencoder to extract image features was constructed in [22]. In [23], the authors integrated a random forest classifier into the stacked sparse autoencoder for hyperspectral image classification, which produced promising generalization performance, prediction accuracy, and operation speed. A dual adversarial autoencoder for image clustering was proposed in [24], which achieved comparable clustering accuracy to CNN models. In [25], an image compression architecture based on energy compression using a convolutional autoencoder was established, which achieved high coding efficiency. In [26], the authors used an autoencoder to project high-dimensional image feature vectors into low-dimensional potential space and proposed an image selection method based on autoencoder neural networks and then applied it to semisupervised image classification. An enhanced fast NSGA-II based on a special congestion strategy and an adaptive crossover strategy, namely ASDNSGA-II, was proposed to improve the selection strategy in [27]. The studies above demonstrate that autoencoders can effectively extract the image features through dimension reduction and show good image recognition results when combined with classification or clustering methods.

3. Methodology

3.1. Methodological Framework. In this study, an autoencoder with multiple hidden layers was used to compress the image features. Based on the compressed low-dimensional data, classification and clustering methods were used for traffic state recognition. The proposed method consisted of four steps (Figure 1), described as follows:

(1) Establishment of image datasets. Traffic images were generated based on traffic surveillance videos. The traffic state was determined manually for each frame, and an image dataset with three traffic states (i.e., free traffic flow, steady traffic flow, and congested traffic flow) was established.

(2) Image data preprocessing. The image matrix was transformed into row vectors, and the vector element
values were normalized and taken as the input of the autoencoder.

(3) Construction of the autoencoder. The influence of the main structural parameters, such as the dimensionality of the input data, the number of hidden layers, and the dimensionality of the dimension-reduced data, on the performance of the autoencoder was evaluated, and as a result, an autoencoder suitable for traffic image processing was constructed. The autoencoder was trained, tested, and optimized based on feedback.

(4) Traffic state recognition. The support vector machine (SVM), deep neural network (DNN), and k-means clustering method were used to classify the feature data obtained using the autoencoder, thereby achieving traffic state recognition.

3.2. Autoencoder Construction. The autoencoder was composed of an encoder and a decoder. The encoder extracted the features of the original image data, and the decoder restored the feature data. The smaller the difference between the restored data and the original data was, the better the autoencoder, and the more effective the extracted feature data.

Traffic surveillance video images were taken as the input of the autoencoder. Both the encoder and decoder had $N$ hidden layers, and the network had a symmetrical structure. There was a fully connected weighting structure between each pair of layers, as shown in Figure 2. Taking the $q^{th}$ image in an image dataset with a sample size $Q$ as an example, the operation of the autoencoder is as follows:

(1) The $q^{th}$ image is converted into a $1 \times 1$ vector

$$X_q^0 = (x_q^0, x_q^1, \ldots, x_q^N),$$  \hspace{1cm} (1)

where $q \in [1, Q]$. Here, $r_0 = l \times h$ and $l, h$ are the width (pixels) and height (pixels) of the image, and $i$ is the element number of $X_q^0$.

(2) Encoding: $N$ encoding hidden layers were used to perform a nonlinear transformation of $X_q^0$ to obtain $N$ encoding vectors $X_q^1, X_q^2, \ldots, X_q^N$, with the dimensions of $1 \times Y_1, 1 \times Y_2, \ldots, 1 \times Y_n, \ldots, 1 \times Y_N$, respectively. The calculation equation is as follows:

$$X_q^0 = \Gamma(w_q^0X_q^0 + z_q^0),$$  \hspace{1cm} (2)

where $\Gamma$ is the encoding nonlinear activation function, which was a hyperbolic tangent function $\tanh$ in this study; $w_q^0$ is the encoding weight matrix of the $n^{th}$ encoding hidden layer in the $q^{th}$ image; $X_q^N$ represents the encoding vector of the $q^{th}$ image; $z_q^0$ represents the encoding offset matrix of the $n^{th}$ encoding hidden layer in the $q^{th}$ image; $n$ represents the encoding hidden layer number, and $n \in [1, N]$.

The calculation equation of $\tanh$ is as follows:

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}},$$  \hspace{1cm} (3)

where $x$ is equal to $w_q^0X_q^0 + z_q^0$.

(3) Image decoding: $N$ decoding hidden layers were used to perform a nonlinear transformation of $X_q^N$, to obtain $N$ decoding vectors, with the dimensions of $\bar{X}_N^q, \ldots, \bar{X}_2^q, \bar{X}_1^q$, respectively. The decoding vectors were restored to the decoding vector $\bar{X}_0^q$ with a dimension of $1 \times r_0$. The calculation equation is as follows:

$$\bar{X}_0^q = \bar{\Gamma}(\bar{w}_q\bar{X}_0^q + \bar{z}_q^0),$$  \hspace{1cm} (4)

where $\bar{\Gamma}$ is the decoding nonlinear transformation function, which was a hyperbolic tangent function.
(4) Feedback optimization of encoding-decoding training: with the least mean square (LMS) as the objective, through continuous feedback propagation training, the autoencoder minimizes the error between the input $X_{q}^{i}$ and output $\tilde{X}_{q}^{i}$, to obtain the optimal autoencoder structure and encoding data. The error loss calculation equation is as follows:

$$\min L(X_{q}^{i}, \tilde{X}_{q}^{i}) = \frac{1}{\tau_{0}} \sum \lvert x_{q}^{i} - \tilde{x}_{q}^{i} \rvert^{2}$$

s.t. $X_{q}^{i} = (x_{q}^{1}, x_{q}^{2}, \ldots, x_{q}^{n}, x_{q}^{t_{0}})$.  

3.3. Traffic State Recognition. In this study, the dimension-reduced data $\tilde{X}_{N}$ obtained using the optimal autoencoder $A^{*}$ was used as the input of the classification and clustering methods for traffic state recognition, including linear classifier in [28], SVM in [29], DNN in [30], and DNN linear combined classifier in [28].

$$\tilde{X}_{N} = (\tilde{X}_{N}^{1}, \tilde{X}_{N}^{2}, \ldots, \tilde{X}_{N}^{q}, \ldots, \tilde{X}_{N}^{Q}).$$  

Here, $\tilde{X}_{N}^{q}$ is the encoding vector of the $q$th image obtained using $A^{*}$, $\tilde{X}_{N}^{q} = (\tilde{x}_{1}^{q}, \tilde{x}_{2}^{q}, \ldots, \tilde{x}_{n}^{q}, \ldots, \tilde{x}_{t_{0}}^{q})$, $\tilde{x}_{i}^{q}$ is the value of the $i$th element, and $Q$ represents the sample size of the dataset, i.e., the number of images.

(1) The linear classifier, a typical supervised learning model, finds the optimal hyperplane that separates the two classes of samples according to the characteristics of the samples. Taking $\tilde{X}_{N}^{q} = (\tilde{x}_{1}^{q}, \tilde{x}_{2}^{q}, \ldots, \tilde{x}_{t_{0}}^{q})$ as an example, let the optimal hyperplane be equation (14).

$$\kappa_{1}\tilde{x}_{1}^{q} + \kappa_{2}\tilde{x}_{2}^{q} + \cdots + \kappa_{n}\tilde{x}_{n}^{q} + \kappa_{t_{0}}\tilde{x}_{t_{0}}^{q} + \kappa_{0} = 0.$$  

The left part of the equation can be defined as a linear discriminant function, as shown in (15).

$$g(\tilde{x}^{q}) = \sum_{i=1}^{n} \kappa_{i}\tilde{x}_{i}^{q} + \kappa_{0}.$$  

Let the weights $\kappa = \{\kappa_{1}, \kappa_{2}, \ldots, \kappa_{n}\}$ and $\kappa_{0}$ be the offset. Then, the inner product is $g(\tilde{x}^{q}) = \kappa^{T}(\tilde{X}_{N}^{q}) + \kappa_{0}$. The linear classifier $g(\tilde{x}^{q})$ divides the feature space into two regions: $g(\tilde{x}^{q}) > 0$ and $g(\tilde{x}^{q}) < 0$; in other words, the data points are divided into two categories: $\sigma_{1}$ and $\sigma_{2}$. The optimal hyperplane can be obtained by training the sample data.

$$g(\tilde{x}^{q}) = \kappa^{T}(\tilde{X}_{N}^{q}) + \kappa_{0} = \begin{cases} > 0, & \tilde{x}^{q} \in \sigma_{1}, \\ < 0, & \text{beyond recognition}, \\ = 0, & \tilde{x}^{q} \in \sigma_{2}. \end{cases}$$  

(2) As a linear classifier, SVM is widely used in image processing, face detection, speech recognition, and so on. The dimension-reduced data that corresponds to each image in $\tilde{X}_{N}$ was labeled with a category label $\pi$, $\pi = \{1, -1\}$, which indicates that the data are to be divided into two categories. Then, a training set $U = \{(\tilde{X}_{N}^{1}, \pi_{1}), (\tilde{X}_{N}^{2}, \pi_{2}), \ldots, (\tilde{X}_{N}^{q}, \pi_{q}), \ldots, (\tilde{X}_{N}^{Q}, \pi_{Q})\}$ was established to train the SVM, thereby obtaining the optimal hyperplane to classify the samples.

$$\mu^{T}\tilde{X}_{N} + b = 0.$$  

Here, $\mu \in R^{n}, b \in R$ represent the normal vector and intercept of the hyperplane, respectively. By introducing a slack variable $\alpha$, the matrix form of the SVM can be expressed as follows:

$$\min_{\mu, b, \alpha} \frac{1}{2} \|\tilde{X}_{N}^{T}\mu + eb\|^{2} + \nu \alpha$$

s.t. $\begin{cases} -(\tilde{X}_{N}^{T}\mu + eb) \geq e - \alpha \\ \alpha > 0. \end{cases}$

where $(\tilde{X}_{N})^{T}$ is the transpose of the matrix $\tilde{X}_{N}$, $\nu$ is the penalty parameter, and $e$ is the vector in which
each element is 1. By solving the optimization problem of (18), μ and can be solved.

(3) The DNN mainly includes an input layer, a hidden layer, and an output layer. There is a fully connected structure of weights between each pair of layers. This method is widely used in image recognition, natural language processing, speech recognition, and so on. Its structure is shown in Figure 3.

The input layer has $Y_N$ neurons, which are connected with the element values of $Y_N$ and correspond to $\mathbf{X}_N = (x_1^N, x_2^N, \ldots, x_i^N, \ldots, x_j^N)$. After being processed by $M$ hidden layers (the dimension of each layer is $1 \times \Omega_1, 1 \times \Omega_2, \ldots, 1 \times \Omega_M$), the probability of the $c$ category is obtained. The sample category is obtained using the softmax function.

(4) The DNN linear combined classifier consisted of two parts: a linear model and a DNN model. The linear model could receive inputs of continuous features or sparse discrete features and quickly converge to effective feature vectors using $L1$ regularization. The principle of the DNN is the same as described above. The DNN Linear combined classifier used the joint training method to simultaneously feedback the error to the linear model and the DNN model to update the parameters. The sigmoid function was used to stack the outputs of the linear model and the DNN model to obtain the final output.

### 4. Field Experiments

#### 4.1. Dataset

The original data consisted of traffic surveillance videos of a main road in a Chinese city, and the image resolution was $704 \times 576$ pixels. The images were extracted and divided into three traffic states: free traffic flow (free), steady traffic flow (steady), and congested traffic flow (congested). Two datasets were constructed: $A1$ has a sample size of 1500, which was used to evaluate the influence of the autoencoder structural parameters, and $A2$ has a sample size of 6000, which was used to optimize the autoencoder and traffic state recognition. The number of images was the same for each traffic state in both datasets. The datasets are shown in Figure 4.

#### 4.2. Optimization of the Autoencoder

##### 4.2.1. Model Training

The training parameters included the basic learning rate, batch training size, and the number of iterations. In the process of model training, the adaptive moment estimation (Adam) iterative optimization algorithm in [31] was used to dynamically adjust the learning rate of each parameter using the first-order moment estimation and second-order moment estimation. The equations are as follows:

$$\chi_t = \eta \times \chi_{t-1} + (1 - \eta) \times g_t, \tag{12}$$

$$\vartheta_t = \rho \times \vartheta_{t-1} + (1 - \rho) \times g_t^2, \tag{13}$$

$$\tilde{\chi}_t = \frac{\chi_t}{1 - \eta^t}, \tag{14}$$

$$\tilde{\vartheta}_t = \frac{\vartheta_t}{1 - \rho^t}, \tag{15}$$

$$v_t = v_{t-1} - \frac{\tilde{\chi}_t}{\sqrt{\tilde{\vartheta}_t} + \epsilon} \times \beta, \tag{16}$$

where $g_t$ is the gradient of time step $t$; $\chi_t$ is the first-order moment estimation of $g_t$, i.e., the exponential moving average of $g_t$; $\chi_t$ is the second-order moment estimation of $g_t$, i.e., the exponential moving average of $g_t^2$; $\eta$ and $\rho$ are the exponential decay rates; $\chi_t$ is the deviation correction of $\chi_t$; $\chi_t$ is the deviation correction of $\chi_t$; $v_t$ is the parameter vector of time step $t$; the default learning rate is $\beta = 0.001$; and $\epsilon$ is the residual term, $\epsilon = 10^{-8}$.

The Adam algorithm has a low requirement for memory during the computation. After deviation correction, each iterative learning rate was within a certain range, which results in stable parameters. Therefore, the Adam algorithm was suitable for training and optimization of the large datasets and high-dimensional data that were addressed in this study.

#### 4.2.2. Evaluation of the Autoencoder

According to the loss distribution curve of the autoencoder, two indexes were used to measure its convergence and evaluate the performance of the autoencoder.

(1) A threshold $\Theta$ was set for the loss. When the training loss of the autoencoder became less than $\Theta$ after performing iterations to step $\theta$, the average loss $T_\Theta$ between steps $\theta - D$ was calculated, as shown in Figure 5. The smaller the value of $T_\Theta$ is, the smaller the difference between the original input data and the restored data of the autoencoder, and the better the performance of the autoencoder.

$$T_\Theta = \frac{1}{D - \theta + 1} \sum_{j=\theta}^{\theta+D} L_j, \tag{17}$$

where $D$ is the total number of training iterations, $L_j$ is the loss of the $j$th iteration, $j \in [\theta, D]$, and $\Theta$ is the loss threshold.

(2) The loss distribution curve of the autoencoder was fitted, and the tangent slope at the end of the fitting curve was calculated.

$$k_D = y'(D), \tag{18}$$

where $y'$ is the loss fitting curve function and $y'(D)$ is the first-order derivative at the $D^{th}$ iteration, i.e., the slope.

As shown in Figure 6, when $k_D$ was less than 0; the closer it is to 0, the gentler the loss curve, and the better the convergence effect.
Under the same parameter settings and training conditions, as $L_\theta$ decreased, more effective feature data could be extracted and restored by the autoencoder; as $k_D$ became less than 0 and approached 0, the convergence effect of the autoencoder loss was better, which indicates high encoding-decoding stability and reliability.

(3) Structure optimization of the autoencoder. To design an autoencoder for video traffic state recognition, 36 sets of crossover tests were conducted based on the dimension of the input data, the dimension of the dimension-reduced data, and the number of hidden layers, as shown in Table 1.
The 36 models were trained using the sample dataset $A_1$ with the same training parameters. The basic learning rate was 0.0001. The batch training size was 100, and the number of iterations was 15,000.

The distribution of the loss was analyzed, and the average loss when $\Theta$ was 0.01 was calculated, as shown in Figure 7. The average losses of the 3rd, 6th, 9th, 12th, 15th, 18th, 21st, and 24th groups were at a lower level compared with other groups of tests. In these eight groups of tests, the input data dimension of the first four groups was $32 \times 32$ and that of the last four groups was $64 \times 64$. The number of hidden layers in each test was 10. Therefore, when the input data dimension decreased and the number of hidden layers increased, $L_{0.01}$ decreased; the effect of the dimension of the dimension-reduced data on the loss was not significant. However, as the dimension of the dimension-reduced data increased, the model training loss tended to decrease. The eight groups of models were preliminarily selected as the candidate models.

Then, curve fitting was performed on the loss of the 36 groups of tests. The coefficients of determination $R^2$ of the fitting functions were all between 0.84 and 0.95. Only 13 groups of tests had $R^2$ lower than 0.9. Therefore, the goodness of fit was high on the whole. The $k_D$ of the loss fitting function of each group of tests was calculated, as shown in Figure 8. In general, the tangent slopes of the loss fitting curves of each group of tests were all less than 0 and close to 0. The curve convergence effect was good, and the tangent slopes of the eight candidate models were all small. Therefore, the autoencoder models of the 3rd, 6th, 9th, 12th, 15th, 18th, 21st, and 24th groups of tests were selected as the candidate models and numbered as model I–model VIII. The structure is shown in Table 2.

Using the sample dataset $A_2$, the candidate models I–VIII were tested. The basic learning rate was 0.0001. The batch training size was 300, and the number of iterations was 100,000. According to the distribution of the training loss, $\Theta$ was set to 0.005. $T_{0.005}$ and $k_D$ of each model were calculated. The results are shown in Table 3. Compared with other models, model III had the smallest value of $T_{0.005}$ and the tangent slope $k_D$. Therefore, given the performance of model III, it was selected as the optimal autoencoder model $A^*$. 

### Table 1: Design of crossover tests.

| Input data dimension | Dimension-reduced data dimension       | Number of hidden layers | Model number |
|----------------------|----------------------------------------|-------------------------|--------------|
| $32 \times 32$      | 3–6, step length: 1                    | 6–10, step length: 2    | 1–12         |
| $64 \times 64$      | 3–6, step length: 1                    | 6–10, step length: 2    | 13–24        |
| $128 \times 128$    | 3–6, step length: 1                    | 6–10, step length: 2    | 25–36        |

### Figure 6: Calculation diagram of $k_D$. 

### Figure 7: $T_{0.01}$ distribution of crossover tests. 

### Figure 8: Tangent slope $k_D$ distribution of loss curve.
5. Result Analysis and Discussion

5.1. Traffic State Recognition and Evaluation Methods.

We assume that there were $C$ types of samples (i.e., traffic states) in the traffic video image dataset. For an arbitrary image sample, its traffic state can be accurately recognized or can be misjudged as other traffic states. Therefore, in the recognition results of certain types of traffic states, there might be errors and omissions. To effectively evaluate the traffic state recognition model, the accuracy rate and recall rate were used to measure the error and omission situations. Furthermore, a comprehensive evaluation index $F_1$ was proposed. The larger the value of $F_1$ is, the better the performance of the model. The principle and calculation process of the three indexes are as follows:

1. The recognition accuracy rate, recall rate, and $F_1$ value of each traffic state in the image datasets were calculated as follows:

$$P_c = \frac{T_c}{S_c}$$

$$R_c = \frac{T_c}{Q_c}$$

$$F_1 = \frac{2 \cdot P_c \cdot R_c}{P_c + R_c}$$

Here, $P_c$ is the total number of traffic states in the traffic image dataset. In this study, $C = 3$ (i.e., free, steady, and congested traffic flow); $c$ is the category of the traffic state, $c \in [1, C]$; $P_c$ is the accuracy rate of the model on the $c^{th}$ type of sample in the image dataset; $R_c$ is the recall rate of the model on the $c^{th}$ type of sample in the image dataset; $F_1$ is the $F_1$ value of the model on the $c^{th}$ type of sample in the image dataset; $T_c$ is the number of samples of the $c^{th}$ type correctly recognized using the model; $S_c$ is the number of samples of the $c^{th}$ type obtained using the model; and $Q_c$ represents the actual number of samples of the $c^{th}$ type in the image dataset.

2. The average accuracy rate, recall rate, and $F_1$ value of $C$-type traffic state all were calculated.

$$\bar{P} = \frac{\sum_{c=1}^{C} P_c}{C}$$

$$\bar{R} = \frac{\sum_{c=1}^{C} R_c}{C}$$

$$\bar{F}_1 = \frac{2 \cdot \bar{P} \cdot \bar{R}}{\bar{P} + \bar{R}}$$

where $\bar{P}$ is the average accuracy rate of the model on the $C$-type traffic state in the image dataset, $\bar{R}$ is the average recall rate of the model on the $C$-type traffic state in the image dataset, and $\bar{F}_1$ is the average $F_1$ value of the model on the $C$-type traffic state in the image dataset.

5.2. $A^*$-Classifier Recognition Results. The traffic images were encoded using the autoencoder $A^*$, from which the dimension-reduced data were obtained and taken as input to the SVM, DNN, linear classifier, and DNN Linear classifier for traffic state recognition. In this study, the models that integrated $A^*$ with the SVM, DNN, linear classifier, and
DNN Linear classifier were named \( A^*\text{-SVM} \), \( A^*\text{-DNN} \), \( A^*\text{-Linear} \), and \( A^*\text{-DNN Linear} \), respectively. Traffic state recognition was conducted on dataset \( A_2 \). The average accuracy rate \( \overline{P} \), recall rate \( \overline{R} \), and \( F_1 \) value all were calculated, as shown in Table 4.

The \( \overline{P} \) and \( \overline{R} \) values of the four models were all in the range of 94.5%–97.1%, and the average \( F_1 \) value was 94.4%–97.1%, which indicate that the integration of the \( A^* \) autoencoder with the classification methods produced excellent traffic state recognition results. Thus, \( A^* \) can effectively extract traffic image features to achieve traffic state recognition.

To further analyze the recognition results of the model, the accuracy rate, recall rate, and \( F_1 \) value of each traffic state all were calculated, and the results are shown in Figure 9. The accuracy rate, recall rate, and \( F_1 \) value varied among the four models. The accuracy rates ranged between 89.0% and 100%. Only the accuracy rate of \( A^*\text{-SVM} \) for the steady traffic flow and that of \( A^*\text{-Linear} \) for the congested traffic flow were below 90%; the recall rates were 86.4%–100%, and specifically, the recall rate of \( A^*\text{-SVM} \) for congested traffic flow and that of \( A^*\text{-Linear} \) for steady traffic flow were below 90%; the \( F_1 \) values ranged between 91.1% and 100%, which indicates that the overall performances of the four models were all good.

From the perspective of traffic states, the accuracy rate, recall rate, and \( F_1 \) value for free traffic flow were the best in all of the models, compared with steady and congested traffic states. The values were close to 100%, which indicates that almost all free traffic flow images were correctly recognized. The accuracy rate and recall rate for steady and congested traffic states were mostly higher than 90%, and the accuracy rate of \( A^*\text{-SVM} \) for steady traffic flow and the recall rate of \( A^*\text{-DNN} \) for congested traffic flow, as well as the accuracy rate of \( A^*\text{-Linear} \) for congested traffic flow and the recall rate of \( A^*\text{-Linear} \) for steady traffic flow, were below 90%. Figures 10–12 show the recognition results of the four models for the three traffic states in dataset \( A_3 \). The autoencoder \( A^* \) showed a good effect on the image dimensionality reduction. Integration of the autoencoder with the SVM, DNN, linear classifier, and DNN Linear classifier achieved overall good recognition results for the different traffic states.

5.4. Comparison between the Proposed Model and CNN Model

The CNN is a typical deep learning method. Common CNN models include AlexNet, VGG16, GoogLeNet, and LeNet in [32], which have been widely applied in the field of image pattern recognition and have achieved remarkable results. In this study, we used dataset \( A_3 \) to train and test the four CNN models, and we calculated the accuracy rate, recall rate, and \( F_1 \) value, and the results of which are shown in Table 6.

Among the four CNN models, AlexNet had the optimal performance; its accuracy rate, recall rate, and \( F_1 \) value all reached approximately 94%. The \( F_1 \) values of the proposed \( A^* \) classifier and \( A^*\text{-k-means} \) clustering model were 94.4%–97.1%, which was higher than that of AlexNet. The performance of LeNet in traffic state recognition was average, with an accuracy rate, recall rate, and \( F_1 \) value of 82.3%, 62.4%, and 71.0%, respectively. Moreover, the \( F_1 \) values of GoogLeNet and VGG16 were all below 40%, which suggests poor performance in traffic state recognition.

From the perspective of the model network structure, \( A^* \) used only five encoding hidden layers to obtain the image feature data, whereas AlexNet, LeNet, GoogLeNet, and VGG16 had 12, 9, 23, and 22 network layers, respectively. The complexity of \( A^* \) was much lower than those models. Therefore, by integrating a simple and practicable autoencoder with common classifiers, the models proposed in this study were able to achieve or surpass the traffic state recognition effect when combined with common clustering methods, which indicates that \( A^* \) can effectively extract the traffic image features.

### Table 4: average values of the evaluation indexes of the A* classifier for each of the traffic states in dataset A_2 (%).

| Model               | \( \overline{P} \) | \( \overline{R} \) | \( F_1 \) |
|---------------------|---------------------|---------------------|----------|
| \( A^*\text{-SVM} \) | 96.2                | 95.8                | 96.0     |
| \( A^*\text{-DNN} \) | 96.2                | 96.1                | 96.2     |
| \( A^*\text{-Linear} \) | 94.5               | 94.5                | 94.4     |
| \( A^*\text{-DNN Linear} \) | 97.1               | 97.1                | 97.1     |

![Figure 9: Classification results for each type of traffic state.](image)
recognition effect of complex CNN networks. Compared with the CNN, the method proposed in this study has the advantages of a lightweight model structure and a low calculation cost, which outperforms CNN models such as AlexNet and GoogLeNet. Therefore, it is suitable for traffic state recognition from videos.

Figure 10: Recognition of A*-SVM, A*-DNN, A*-Linear, and A*-DNN_Linear for free traffic flow ($F_1 = 100\%, 100\%, 99.1\%, 100\%)$. 

Figure 11: Recognition of A*-SVM, A*-DNN, A*-Linear, and A*-DNN_Linear for steady traffic flow ($F_1 = 94.1\%, 94.3\%, 91.1\%, 95.6\%)$. 
Figure 12: Recognition of $A^*$-SVM, $A^*$-DNN, $A^*$-Linear, and $A^*$-DNN_Linear for congested traffic flow ($F_1 = 93.4\%, 94.5\%, 92.6\%, 95.7\%$).

Table 5: Evaluation indexes of $A^*$-k-means for each traffic state in dataset $A_2$ (%).

| Traffic state | Accuracy rate | Recall rate | $F_1$ value |
|---------------|---------------|-------------|-------------|
| Free          | 90.7          | 95.6        | 93.1        |
| Steady        | 100           | 100         | 100         |
| Congested     | 95.3          | 90.2        | 92.7        |
| Average       | 95.4          | 95.3        | 95.3        |

Table 6: Test results of the CNN models and the models proposed in this paper.

| Model         | Model structure                                                                 | Accuracy rate (%) | Recall rate (%) | $F_1$ (%) |
|---------------|----------------------------------------------------------------------------------|-------------------|-----------------|-----------|
| AlexNet       | 5 convolutional layers, 3 pooling layers, 3 fully connected layers, and 1 classification layer | 94.5              | 93.6            | 94.0      |
| LeNet         | 3 convolutional layers, 2 pooling layers, 1 fully connected layer, and 1 classification layer | 82.3              | 62.4            | 71.0      |
| GoogLeNet     | 22 network layers, including convolutional layers and pooling layers, and 1 classification layer | 36.8              | 35.2            | 36.0      |
| VGG16         | 13 convolutional layers, 3 fully connected layers, 5 pooling layers, and 1 classification layer | 11.1              | 33.3            | 16.7      |
| $A^*$ Classifier | 5 encoding hidden layers + common classifiers (linear classification, SVM, DNN, and so on) | 94.5–97.1         | 94.5–97.1       | 94.4–97.1 |
| $A^*$-k-means | 5 encoding hidden layers + k-means clustering                                      | 95.4              | 95.3            | 95.3      |
6. Conclusions
In this study, an autoencoder model for urban traffic surveillance videos was proposed that can effectively reduce the data dimension by optimizing the input data dimension, number of hidden layers, and dimension of the dimension-reduced data. Taking the low-dimensional image features obtained using the autoencoder as the input, five models were constructed based on common classification methods, including the linear classifier, DNN, SVM, DNN Linear, and k-means clustering method. The performances of the models in traffic state recognition were compared with those of commonly used CNN models. The results show that the average $F_1$ value of the four models A*-DNN, A*-SVM, A*-Linear, and A*-DNN Linear was 94.4%–97.1%, and the average $F_1$ value of A*-k-means was 95.3%. Among the CNN models, AlexNet has the best performance, with an $F_1$ value of 94.0%. Thus, the autoencoder constructed in this study can effectively extract the image features. When integrated with common classification and clustering methods, it can accurately recognize the traffic state, and it achieves better results than common CNN models, such as AlexNet, LeNet, GoogLeNet, and VGG16.

The traffic state recognition model was established in two stages to avoid the problems of high algorithm complexity and calculation cost. First, an autoencoder was proposed which effectively reduces video dimension; then, the traffic state recognition model was established by integrating the autoencoder with common classification and clustering methods. The train and test result presents that our method has advantages of a lightweight model structure and a low calculation cost, which outperforms CNN models such as AlexNet and GoogLeNet. The method can also be applied in other fields of video detection, such as image compression, moving target detection, and image classification.

Due to the variability and complexity of traffic scenes and the difficulty in building large-scale traffic image datasets, we conducted an exploratory study on the construction and optimization of an autoencoder model using limited traffic scenes and sample size. In the future, we will focus on the train and test dataset construction, autoencoder model architecture and parameter optimization, and classification method selection.

Data Availability
The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest
The authors declare that there are no conflicts of interest.

Acknowledgments
This research was supported in part by the National Natural Science Foundation of China under grant 61703064, Chongqing Research Program under grant cstc2018jcxmsybX0295, and Scientific Research Project of Chongqing Key Lab of Traffic System & Safety in Mountain Cities under grant 2018TSSMC05.

References
[1] R. F. Quiros, R. A. Bedruz, A. C. Uy, A. Abad, A. Bandala, and E. P. Dadios, “Machine vision of traffic state estimation using fuzzy logic,” in Proceedings of the IEEE Region 10 Conference, Singapore, 2016, February.
[2] H. Cui, C. Yuan, Z. F. Wei et al., “Traffic state recognition using state images and FCM,” Journal of Xidian University, vol. 6, pp. 85–90, 2017.
[3] L. Lin, J. Hu, Q. Huang, and J. Y. Song, “A fuzzy hidden Markov model for traffic status classification based on video features,” in Proceedings of the IEEE Conference on Computational Engineering in Systems Applications, Beijing, China, 2007, July.
[4] H. Hu, Z. Gao, Y. Sheng, C. Zhang, and R. Zheng, “Traffic density recognition based on image global texture feature,” International Journal of Intelligent Transportation Systems Research, vol. 17, no. 3, pp. 171–180, 2019.
[5] Z. H. Zhang, F. Min, G. S. Chen, S. P. Shen, Z. C. Wei, and X. B. Zhou, “Tri-partition state alphabet-based sequential pattern for multivariate time series,” Cognitive Computation, pp. 1–19, 2021.
[6] X. Ran, X. Zhou, M. Lei, W. Tepsan, and W. Deng, “A novel k-means clustering algorithm with a noise algorithm for capturing urban hotspots,” Applied Sciences, vol. 11, no. 23, Article ID 11202, 2021.
[7] Y. Yuan, W. Zhang, X. Yang, Y. Liu, Z. Liu, and W. Wang, “Traffic state classification and prediction based on trajectory data,” Journal of Intelligent Transportation Systems, vol. 15, pp. 1–15, 2021.
[8] H. Cui, Y. Guan, H. Chen, and W. Deng, “A novel advancing signal processing method based on coupled multi-stable stochastic resonance for fault detection,” Applied Sciences, vol. 11, no. 12, p. 5385, 2012.
[9] D. Roy, K. Naveen Kumar, and C. Krishna Mohan, “Defining traffic states using spatio-temporal traffic graphs,” in Proceedings of the 23rd International Conference on Intelligent Transportation Systems (ITSC), IEEE, Rhodes, Greece, 2020, September.
[10] D. Xu, C. Wei, P. Peng, Q. Xuan, and H. Guo, “GE-GAN: a novel deep learning framework for road traffic state estimation,” Transportation Research Part C: Emerging Technologies, vol. 117, Article ID 102635, 2020.
[11] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” Computer Science, 2014.
[12] C. Szegedy, Y. Wei, P. Sermanet et al., “Going deeper with convolutions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, June 2015.
[13] W. Yong-dong, X. Dong-wei, P. Peng, L. Yi, Z. Gui-jun, and X. Xiao, “Kernel PCA for road traffic data non-linear feature extraction,” IET Intelligent Transport Systems, vol. 13, no. 8, pp. 1291–1298, 2019.
[14] J. Zhai, S. Zhang, J. Chen, and Q. He, “Autoencoder and its various variants,” in Proceedings of the 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Miyazaki, Japan, October 2018.
[15] S. Chen, A. Chang, X. S. Li, and J. Wang, “Development of a road traffic state identification method based on image texture features,” in Proceedings of the Advanced Information
[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[17] A. Nassim, A. Haikel, B. Yakoub, B. Bilel, A. Naif, and Z. Mansour, "Deep learning approach for car detection in UAV imagery," *Remote Sensing*, vol. 9, no. 4, pp. 316–326, 2017.

[18] Y. Koga, H. Miyazaki, and R. Shibasaki, “A CNN-based method of vehicle detection from aerial images using hard example mining,” *Remote Sensing*, vol. 10, no. 1, 2018.

[19] L. Suhao, L. Jinzhao, L. Guoquan, B. Tong, W. Huiqian, and P. Yu, “Vehicle type detection based on deep learning in traffic scene,” *Procedia Computer Science*, vol. 131, pp. 564–572, 2018.

[20] L. Van Der Maaten, E. Postma, and J. Van den Herik, “Dimensionality reduction: a comparative review,” *Mach Learn Res*, vol. 10, pp. 66–71, 2009.

[21] J. Wang, H. He, and D. V. Prokhorov, “A folded neural network autoencoder for dimensionality reduction,” *Procedia Computer Science*, vol. 13, pp. 120–127, 2012.

[22] S. Gao, Y. Zhang, K. Jia, J. Lu, and Y. Zhang, "Single sample face recognition via learning deep supervised autoencoders," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 10, pp. 2108–2118, 2015.

[23] "Spectral-spatial classification of hyperspectral imagery based on stacked sparse autoencoder and random forest," *European Journal of Remote Sensing*, vol. 50, no. 1, pp. 47–63, 2017.

[24] P. Ge, C. X. Ren, D. Q. Dai, J. Feng, and S. Yan, “Dual adversarial autoencoders for clustering,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 99, pp. 1–8, 2019.

[25] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, “Energy compaction-based image compression using convolutional autoencoder,” *IEEE Transactions on Multimedia*, vol. 22, no. 4, pp. 860–873, 2020.

[26] T. Singh, A. K. Gaurav, and F. Ghourabi, “Image selection based on autoencoder neural network and application to the semi-supervised image classification,” in *Proceedings of the the 34th Annual Conference of the Japanese Society for Artificial Intelligence*, Kumamoto, Japan, 2020, March.

[27] W. Deng, X. X. Zhang, Y. Q. Zhou et al., "An enhanced fast non-dominated solution sorting genetic algorithm for multiobjective problems," *Information Sciences*, vol. 585, pp. 441–453, 2022.

[28] H. T. Cheng, L. Koc, J. Harmsen et al., “Wide & deep learning for recommender systems,” in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. Presented at Conference on Recommender Systems*, New York, America, 2016.

[29] Z. Sun, Z. Li, and Y. Zhao, “Traffic congestion forecasting based on possibility theory,” *International Journal of Intelligent Transportation Systems Research*, vol. 14, no. 2, pp. 85–91, 2016.

[30] Y. Bengio, “Learning deep architectures for AI,” *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[31] J. B. Kingma, “A method for stochastic optimization,” *Computer Science*, 2014.

[32] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.