Hindawi
Complexity
Volume 2021, Article ID 5548476, 17 pages
https://doi.org/10.1155/2021/5548476

Research Article

Research on Fast Pedestrian Detection Algorithm Based on Autoencoding Neural Network and AdaBoost

Hongzhi Zhou and Gan Yu

College of Information Engineering, Fuyang Normal University, Fuyang, Anhui 236041, China

Correspondence should be addressed to Gan Yu; yugan@fynu.edu.cn

Received 6 January 2021; Revised 4 March 2021; Accepted 17 March 2021; Published 27 March 2021

Academic Editor: Wei Wang

In order to solve the problem of low accuracy of pedestrian detection of real traffic cameras and high missed detection rate of small target pedestrians, this paper combines autoencoding neural network and AdaBoost to construct a fast pedestrian detection algorithm. Aiming at the problem that a single high-level output feature map has insufficient ability to express pedestrian features and existing methods cannot effectively select appropriate multilevel features, this paper improves the traditional AdaBoost algorithm structure, that is, the sample weight update formula and the strong classifier output formula are reset, and the two-input AdaBoost-DBN classification algorithm is proposed. Moreover, in view of the problem that the fusion video is not smoothly played, this paper considers the motion information of the video object, performs pixel interpolation by motion compensation, and restores the frame rate of the original video by reconstructing the dropped interframe image. Through experimental research, we can see that the algorithm constructed in this paper has a certain effect.

1. Introduction

Pedestrian detection is an enduring research direction in the field of computer vision, and it is an extremely practical subproblem in the big problem of target detection. Some of its applications have a great impact on our daily lives and can be directly applied to indoor and outdoor mobile robots, auto-driving cars, security monitoring, and other scenarios [1].

Pedestrian detection technology can detect pedestrians in front of the vehicle in time and take corresponding measures, which makes this technology a vital research content in vehicle-assisted driving technology. At present, through the combination with other technologies, automatic vehicle driving technology has been realized. This technology cannot only ensure the safety of traffic but also save people from the work of driving a car. The research of vehicle driving technology has been favored by more and more research institutes, universities, and companies at home and abroad and has become the focus of common concern in industry and academia [2].

Pedestrian detection technology enables intelligent robots to view people around, analyze their behaviors, and respond to human instructions so that they can better serve humans. Pedestrian detection technology is not only applied in the fields of intelligent monitoring, vehicle-assisted driving, and intelligent robots, but as humans continue to research and explore intelligence, it can be applied to any field where people appear and need to provide services. In particular, in recent years, the Internet and other media have rapidly transitioned from text images to videos, which has rapidly increased the amount of information available for video. As the most concerned part of the video sequence, pedestrian detection in the video sequence has also become an indispensable and important task in the field of machine vision research and has a pivotal position for subsequent target tracking and behavior analysis [3].

2. Related Work

Pedestrian detection technology, as a research hotspot and focus in the field of computer vision, has attracted the

attention of many research institutions at home and abroad. Through research, domestic and foreign researchers have proposed a large number of excellent pedestrian detection algorithms and obtained good research results. The literature [4] uses the template matching method to establish a layered human body template for the target to be detected, then compares and matches each area of the image to be detected, and judges whether it is the required detection target according to its final matching similarity. However, the implementation process of this algorithm is complicated in calculation and is easily affected by external factors such as environment, so it has not been greatly developed and applied. The literature [5] combined Harr features and SVM classifiers to realize pedestrian detection. The literature [6] proposed an epoch-making image histogram of oriented gradient (HOG). This method detects pedestrians through the combination of the HOG feature of the image and a single SVM classifier, which has a good detection effect. The literature [7] proposed a pedestrian detection algorithm based on HOG features and edge symmetry based on HOG features. This method first filters out the possible pedestrian areas in the image based on the symmetry characteristic of the vertical edges of pedestrians in the image and then uses the HOG + SVM method to detect pedestrians in this area, which improves the detection rate to a certain extent. The literature [8] used the image's multilevel local binary (LBP) and multilevel gradient orientation feature (HOG) as feature sets to detect the head and shoulders of the human body in the image to reduce the dimensionality of the multilevel feature set and achieve accurate statistics on the number of people in complex scenes. With the rapid development of deep learning and other related theories, methods such as artificial god networks have gradually been applied to pedestrian detection. The literature [9] replaced the sliding window detection method in the traditional algorithm with a neural network method and accurately classified the image features to be detected, thereby determining the area to be detected. The currently widely used pedestrian detection methods can be roughly summarized into three types: motion-based analysis method, template-based matching method, and statistics-based learning method [10]. The motion-based analysis method mainly uses the difference in image information of moving objects in different frames of images and performs differential operations on corresponding pixels in adjacent frames of images in the video sequence. After that, it compares the difference result with a preset threshold, obtains the motion information in the video sequence according to the final comparison result, and then uses the characteristic information of the motion area to further detect whether the object is a pedestrian [11]. The template-based matching method is to define a hierarchical human body template for the pedestrian target to be detected, then compare and match the image to be detected with each area of the template, and finally determine whether the target to be detected is a pedestrian based on its matching similarity [12].

The method based on statistical learning refers to obtaining a classifier representing pedestrians by training the image sample data and then using the classifier to judge and classify the image to be detected [13]. Statistics-based methods can be divided into support vector machine (SVM), AdaBoost, and neural network-based methods according to their classifiers. The classifier used in the support vector machine-based method is SVM. SVM is based on the principle of risk minimization. The main idea is to determine a linear function through which the new sample data can be correctly separated in the maximum interval hyperplane [14]. AdaBoost is an iterative algorithm. The main idea of this algorithm when implementing pedestrian detection is to combine different weak classifiers trained from the same set of sample data to form an ultimate strong classifier and then use the strong classifier to detect and search the input image to be detected to determine whether there are pedestrians in the image [15]. The neural network is a research hotspot in the field of machine vision and has been widely used. Its application in the field of pedestrian detection is mainly by extracting high-latitude image features in the image and classifying and recognizing pedestrians in the image according to the features. With the continuous development of computer hardware, combined with the relevant research results of pedestrian detection theory, in the actual development and application process, domestic and foreign researchers and institutions have proposed a series of design schemes [16]. The literature [17] used algorithms such as mixture Gaussian background modeling and moving target detection and tracking. Moreover, it used CCS compiler optimization, software pipeline optimization, algorithm code optimization, TI-related function library, and other optimization methods to implement a complete video target tracking system on the DM6437 hardware platform. The literature [18] designed an effective pedestrian detection system on the DM6437-embedded DSP platform by analyzing the operating cycle of related modules and combining the characteristics of the hardware platform. The literature [19] used the DM8168 hardware platform to propose a pedestrian classification and detection algorithm combining the foreground enhancement detection algorithm with the CENTRIST operator, which realizes the effective detection of the pedestrian detection algorithm on the DSP platform. The literature [20] designed a detection system for pedestrians waiting to cross the road and developed and implemented it on the TMS320C40 platform. The experimental results proved that it has a good detection rate. Based on the TMS3206455 platform, the literature [21] proposed an algorithm based on wavelet pyramid decomposition, combined with the tunnel filtering algorithm to achieve multitarget detection of cars, pedestrians, and bicycles in complex scenes. The literature [22] designed a pedestrian detection hardware platform based on DM6437 and studied a DSP car-assisted driving pedestrian automatic detection system that can detect pedestrians and issue warnings, which has a high detection efficiency.

3. Basic Theory of Texture Feature Extraction

The texture feature describes the phenomenon that the image is repeated or changed in space and reflects the uniformity of the image. It is a feature description consistent

with the human visual perception system. The texture feature can describe the detailed information and change the trend of the local area of the image, so it is effectively used in the field of image classification.

Local Binary Pattern (LBP) is mainly used for texture classification. Because of its simple calculation, it is further used for target tracking, facial expression recognition, medical imaging, and image classification. This method considers the rectangular square in the image and encodes the intensity difference between the center pixel and the N neighborhood pixels on the circle of radius R in the binary format. Then, it assigns a binary value (0 or 1) to each neighboring pixel based on the intensity difference and finally performs a weighted summation to obtain the LBP value of the center pixel. In a similar way, each pixel value in the image is replaced with an LBP value. The calculation formula of LBP is as follows:

$$\begin{aligned} LBP(N,R) &= \sum_{i=1}^{N} 2^{i-1} \times D_1(I_i, I_c), \\ D_1(I_i, I_c) &= \begin{cases} 1, & I_i \geq I_c \\ 0, & I_i < I_c \end{cases}. \end{aligned} \tag{1}$$

In the above formula, R and N represent the radius and the number of neighboring pixels, respectively. I_c represents the center pixel, and I_i represents the neighborhood pixel. An example of LBP calculation is shown in Figure 1, where $R = 1$ and $N = 8$.

For the adjacent pixel $I_n (n = 1, 2, \cdots, 8)$ of the center pixel I_c, the LNDP mode value is calculated according to the following process:

$$\begin{aligned} k_1^n = I_8 - I_n, k_2^n = I_{n+1} - I_n, \quad n = 1, \\ k_1^n = I_{n-1} - I_n, k_2^n = I_{n+1} - I_n, \quad n = 2, 3, \cdots, 7, \\ k_1^n = I_{n-1} - I_n, k_2^n = I_1 - I_n, \quad n = 8. \end{aligned} \tag{2}$$

The difference between each neighboring pixel and the neighboring pixel is k_1^n and k_2^n, and each neighboring pixel is assigned a binary value F based on these two difference values:

$$F(k_1^n, k_2^n) = \begin{cases} 1, & \text{if } k_1^n \geq 0 \text{ and } k_2^n \geq 0 \\ 1, & \text{if } k_1^n < 0 \text{ and } k_2^n < 0 \\ 0, & \text{if } k_1^n \geq 0 \text{ and } k_2^n < 0 \\ 0, & \text{if } k_1^n < 0 \text{ and } k_2^n \geq 0 \end{cases}. \tag{3}$$

For the center pixel I_c, the following binary values can be used to calculate the LNDP mode value:

$$\text{LNDP}(I_c) = \sum_{n=1}^{8} 2^{n-1} \times F(k_1^n, k_2^n). \tag{4}$$

Figure 2 shows the calculation process of the LNDP mode and windows. Figures 2(a) and 2(b) show the number and pixel intensity of the neighboring pixels, and window $(f - m)$ represents the calculation of the LNDP mode for each neighboring pixel. Among them, window (f) means that pixel I_1 calculates the difference between pixel I_1 and pixel I_2 and pixel I_8 through formula (2), and the difference is "1" and "5," respectively. Since the two difference values

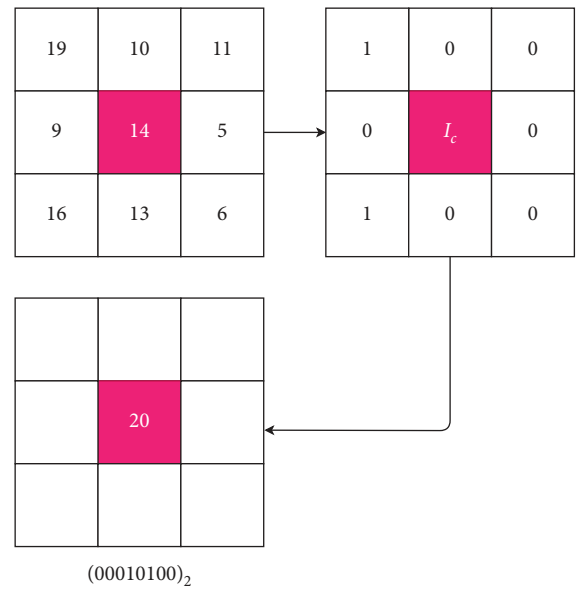

Figure 1: LBP mode.

are both positive numbers, "1" is assigned to the pixel I_1 by formula (3). Similarly, for other pixels, the mode value has been obtained and shown in window Figure 2(c). As shown in window Figure 2(d), the mode value is multiplied by the weight, and the LNDP value is obtained by summing the mode values in window Figure 2(e).

The traditional LBP mode compares the center pixel in the image with the neighboring pixels to generate a binary mode value. This mode only considers the size relationship between pixels, not the difference between pixels. Figure 3 shows two completely different partial structure modes, and the same binary code is obtained after LBP encoding. It shows that the local structure mode encoded by LBP loses a lot of local information, and LBP also ignores the influence of neighboring pixels on its binary encoding.

The local neighborhood enhancement mode calculates the LNIP mode value, as shown in Figure 4. The figure shows the neighborhood (pixel)-adjacent (pixel) relationship of each of the 8 neighborhood pixels of the center pixel I_c. When $i = 1, 3, 5,$ and 7, I_i has 4 adjacent pixels. When $i = 2$, 4, 6, and 8, I_i has 2 adjacent pixels. Its mathematical definition is as follows:

$$S_i = \begin{cases} \left\{ I_{1+\text{mod}(i+5,7)}, I_{1+\text{mod}(i+6,9)}, I_{i+1}, I_{\text{mod}(i+2,8)} \right\}, & i = 1, 3, 5, \text{ and } 7 \\ \left\{ I_{i-1}, I_{\text{mod}(i+1,8)} \right\}, & i = 2, 4, 6, \text{ and } 8 \end{cases}. \tag{5}$$

For the symbol mode of LNIP, we first calculate the relative difference symbol $B_{1,i}$ between the neighboring pixel I_i of the center pixel and its corresponding neighboring pixel S_i. The M bit pattern can be obtained, and M is the number of elements, as shown in formula (6). Similarly, the relative difference symbol $B_{2,i}$ between the central pixel I_c and its corresponding neighboring pixel S_i is calculated, as shown in the following formula:

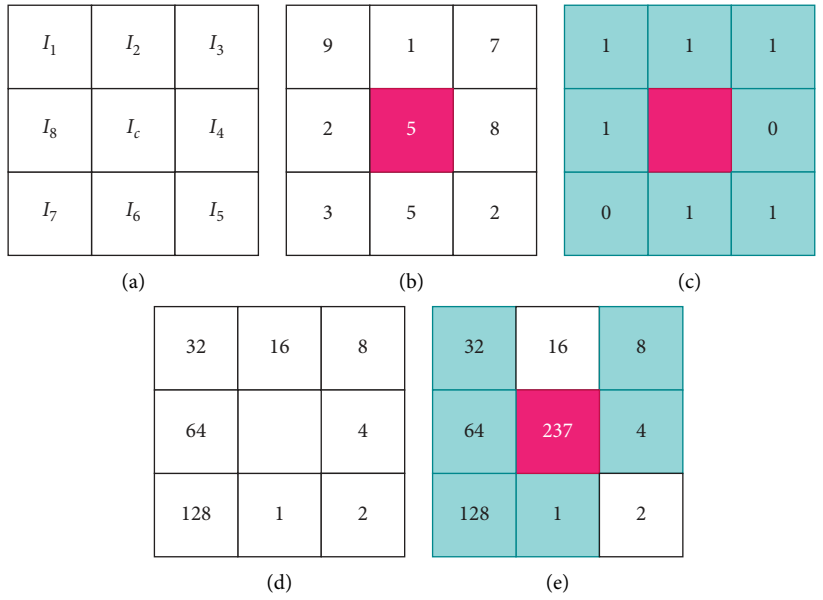

Figure 2: Local neighborhood difference mode.

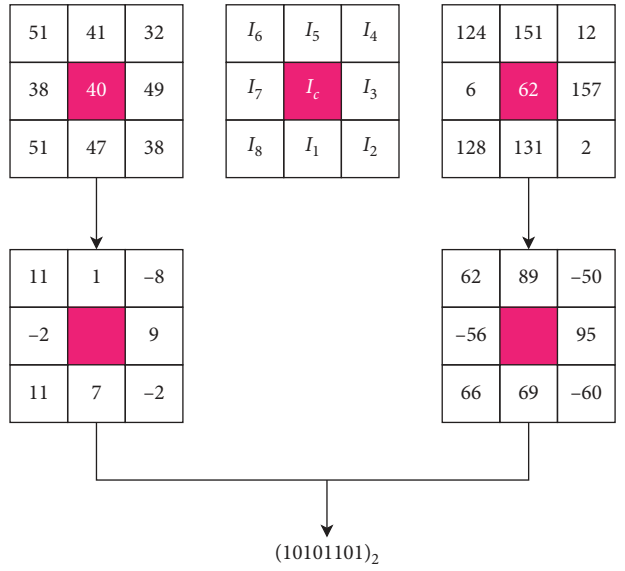

Figure 3: LBP encoding.

$$\begin{aligned} B_{1,i} &= \operatorname{sign}\left(S_i(k), I_i\right), \quad k = 1, \cdots, M, \\ B_{2,i} &= \operatorname{sign}\left(S_i(k), I_c\right), \quad k = 1, \cdots, M, \\ \operatorname{sign}(a,b) &= \begin{cases} 1, & a \geq b \\ 0, & a < b \end{cases}. \end{aligned} \tag{6}$$

By performing bitwise XOR operation on $B_{1,i}$ and $B_{2,i}$, the structure change of the bit pattern is calculated. When calculating the bitwise XOR, the M bit mode value can be obtained by formula (7). The symbol $\#(D_i = 1)$ represents the number of 1 in the binary number D_i, and formula (7) finally determines the final binary value of the 3×3 window. Among them, when i is an odd number, $M = 4$, and when i is an even number, $M = 2$. The symbol mode value of LNIP can be obtained by the following formula:

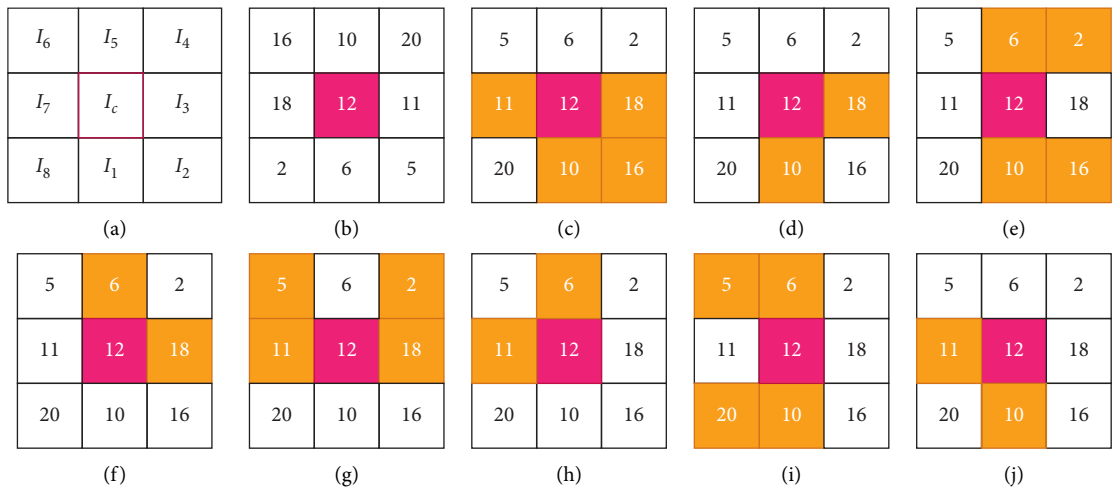

Figure 4: Local neighborhood enhancement mode calculation.

$$D_i = \mathrm{XOR}\left(B_{1,i}, B_{2,i}\right),$$

$$B\left(I_i, I_c\right) = \begin{cases} 1, & \#\left(D_i = 1\right) \geq \frac{1}{2}(M), \\ 0, & \#\left(D_i = 1\right) < \frac{1}{2}(M), \end{cases} \quad \mathrm{LNDP}_s\left(I_c\right) = \sum_{i=1}^{8} 2^{i-1} \times B\left(I_i, I_c\right). \tag{7}$$

When calculating the amplitude mode of the 3×3 window, the concept of statistical dispersion is taken into consideration. To calculate this dispersion, the absolute average deviation of a particular pixel is considered and compared with the average deviation of the center pixel. Specifically, the average deviation M_i of the neighboring pixel $I_i\,(i = 1, 2, \cdots, 8)$ and the corresponding neighboring pixel S_i and the average deviation T_c of the neighboring pixel I_i relative to the center pixel I_c are calculated. Then, the average deviation M_i is compared with the threshold T_c to determine the binary bit value of the adjacent pixel I_i in the 3×3 window. The calculation formula is as follows:

$$M_i = \frac{1}{M} \sum_{k=1}^{M} \left| S_i(k) - I_i \right|,$$

$$T_c = \frac{1}{8} \sum_{i=1}^{8} \left| I_i - I_c \right|, \tag{8}$$

$$\mathrm{LNIP}_M\left(I_c\right) = \sum_{i=1}^{8} 2^{i-1} \times \mathrm{sign}\left(M_i, T_c\right).$$

Figure 3 shows two very different 3×3 pixel blocks through an example, and the same LBP code is obtained through the LBP mode.

In Figure 4, Figures 4(a) and 4(b) represent the distribution of the center pixel and the neighboring pixels. Figures 4(c)–4(j) represent the calculation process of the symbol mode and amplitude mode of the local neighborhood enhancement mode.

4. AdaBoost Algorithm

In the AdaBoost algorithm process, the original dataset is first initialized, that is, each sample is weighted and averaged and assigned a corresponding weight, and the weight of each sample is used to calculate the next algorithm iteration. If a sample is misclassified in the classification, the weight of the sample will increase; otherwise, the weight will decrease. Through this mechanism, the AdaBoost algorithm will focus on solving difficult-to-classify samples. Figure 5 is the schematic diagram of the AdaBoost algorithm. First, the training set is initialized with weights, and weak learner 1 is trained, and the weight of the sample is updated according to the classification error of weak learner 1. That is, the weight of correctly classified samples decreases, and the weight of incorrectly classified samples increases. In this way, misclassified samples get more attention in subsequent iterations. After that, it is carried out in sequence according to the same method until the end of the iteration, and the weak learners are integrated to form a strong learner.

The core idea of the algorithm is to linearly weight several homogeneous classifiers to form a strong classifier, so the algorithm is mainly used to calculate the weight of each classifier. That is, after a weak classifier is given, the initial

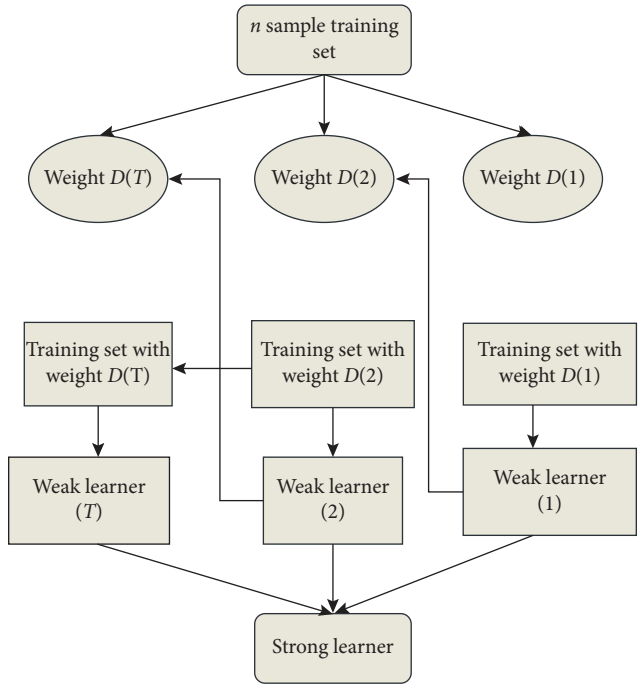

Figure 5: AdaBoost algorithm flow chart.

dataset is first fitted to the weak classifier, the weight of the classifier is obtained according to the fitting result, and the sample weight of the dataset is adjusted.

The AdaBoost algorithm process is as follows:

(1) Train sample weight: in the case of the initial training sample without any prior knowledge, the weight of each sample in N training samples is $1/N$, and D_1 represents the weight set:

$$D_1 = \left(\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}\right). \tag{9}$$

(2) Calculate the classifier error: the weight of the classifier is calculated by the accuracy of each classification. The higher the classification accuracy, the greater the weight of the classifier. In the following formula, $h_t(x_i)$ represents the classification category of sample x_i in the tth iteration, and y_i represents the true label of sample x_i:

$$\varepsilon_t = \sum_{i=1}^{N} D_i^t \left[h_t(x_i) \neq y_i\right]. \tag{10}$$

(3) Calculate the weight of the classifier: according to the classification error obtained in step (20), the proportional coefficient of the weak classifier in the final strong classifier is determined as follows:

$$\alpha = \frac{1}{2} \lg \frac{1 - \varepsilon^{(t)}}{\varepsilon^{(t)}}. \tag{11}$$

(4) Update sample weight: adjust the weight of the sample according to the classification result in step (2). In the following formula, D_i^{t+1} represents the weight of the ith sample in the $t+1$ iteration:

$$D_i^{t+1} = D_i^t \cdot \exp\left(\alpha_t \cdot \left[h_t(x_i) \neq y_i\right]\right). \tag{12}$$

(5) Output strong classifier: the weight of the classifier and the corresponding classification accuracy are linearly superimposed to form a strong classifier, as shown in the following formula:

$$H(x) = \sum_{t=1}^{T} \alpha(t) \left[h(x_i) = y_i\right]. \tag{13}$$

As an improved boosting algorithm, the AdaBoost algorithm has a high integration capability. For a variety of homogeneous classifiers, the algorithm can build a good and strong classification model, and the algorithm will not be overfitting after repeated training.

As a representative of boosting algorithms, AdaBoost has been widely used. Because the AdaBoost algorithm has too strict requirements on the classification accuracy of the base classifier, the algorithm does not have universal applicability. This section improves the algorithm based on this shortcoming. In addition, combined with the base classifier DBN algorithm, the strong classifier formula output in the AdaBoost algorithm is reset.

Aiming at the problem that the AdaBoost algorithm is too strict for the classification accuracy of weak classifiers, Zhu et al. proposed the SAMME algorithm. The calculation

method of the classifier weight α in this algorithm is different, as shown in the following formula:

$$\alpha = \frac{1}{2}\lg\frac{1-\varepsilon^{(t)}}{\varepsilon^{(t)}} + \lg(M-1). \tag{14}$$

The weight of the classifier adds a positive term $\lg(M-1)$ on the original basis, where M represents the total number of categories. The increase of this item has brought a great improvement to the performance of the algorithm. The original AdaBoost algorithm requires the correct rate of weak classifiers to be greater than 1/2, but it is difficult for general classifiers to meet this requirement. The improved classifier weights show that only the classification error $\varepsilon(t) < 1 - (1/M)$ of the classifier is required, that is, as the number of iterations increases, the accuracy requirement of the classifier becomes lower and lower. At the same time, according to the update rules of sample weights, the SAMME algorithm will obtain larger weights for misclassified samples when the sample weights are updated. When $M = 2$, the SAMME algorithm is the same as the AdaBoost algorithm.

It can be seen from the above analysis that the regular term $\lg(M-1)$ added to the calculation formula of the classifier weight α greatly increases the universality of the AdaBoost algorithm. However, this regular term is not arbitrarily given, but is obtained from the forward superposition model of multiple types of exponential loss functions. The expression of the two-class index loss function is as follows:

$$L(k,h) = e^{-kh(x)}. \tag{15}$$

In the above formula, $k \in \{-1, 1\}$, and h is the classification label of the classifier.

The above formula is extended to the expression of the multiclass index loss function, as follows:

$$L(k,h) = \exp\left[-\frac{1}{M}\left(k_1h_1+,\cdots,+k_mh_m\right)\right] = \exp\left[-\frac{1}{M}K^Th\right]. \tag{16}$$

In the above formula, the value of $K = (k_1, k_2, \cdots, k_m)^T$ is

$$k_m = \begin{cases} 1, & h = m, \\ -(1/(M-1)), & h \neq m. \end{cases} \tag{17}$$

In the formula, k_m is the output value of the mth category determined by the classifier, and M is the total number of categories in the dataset. Here, the condition of k needs to be restricted, and the constraint condition $k_1 + k_2 + , \cdots, +k_m = 0$ is added according to the above formula. In this case, when $M = 2$, the m-class index loss function becomes a two-class index loss function.

The design of the ensemble model uses the DBN classifier as the base classifier. According to the meaning of the DBN structure to output the classification results, the formula for the output strong classifier $H(x)$ is improved, as shown in the following formula:

$$H(x) = \sum_{t=1}^{T}\alpha(t)h'(x). \tag{18}$$

The original formula is

$$H(x) = \arg\max_k \sum_{t=1}^{T}\alpha(t)[h(x) = k]. \tag{19}$$

The original formula is decomposed and analyzed: the algorithm symbol $[a]$ in $[h(x) = k]$ can be interpreted as that when the logical expression a is true, $[a] = 1$; otherwise, $[a] = 0$. Therefore, the output form of the formula $[h(x) = k]$ is a matrix composed of 0 and 1. 1 means the classification is correct, and 0 means the classification is wrong. Finally, using the argmax function, the maximum value of each row is obtained for each sample category. The improvements made in this article are

$$H(x) = \sum_{t=1}^{T}\alpha(t)h(x), \tag{20}$$

where $h(x)$ represents the proportion of each sample in each category before DBN output, and it is usually a decimal. At this time, the weight of the classifier weights the original data, which can better reflect the results of each classification.

The DBN network structure consists of four parts: the number of input layer nodes, the number of output layer nodes, the number of hidden layers, and the number of hidden layer nodes. This article uses the following empirical formula to determine network parameters:

$$S = \sqrt{mn} + \frac{k}{2}. \tag{21}$$

Among them, S is the number of hidden layer nodes, m is the number of input layer nodes, n is the number of output layer nodes, and k is a constant between 1 and 10. From this empirical formula, the value of the number of hidden layer nodes can be obtained, and the final number of hidden layer nodes can be determined through experiments. Since m and n are known, the number of hidden layer nodes S can be determined by the value of k. According to the above formula, there are p values for the number of first layer nodes S_1 in the hidden layer:

$$S_1 = \left[S_{11}, S_{12}, \cdots, S_{1p}\right]. \tag{22}$$

When the number of hidden nodes in the first layer is determined, it is used as the input layer to determine the number of nodes in the second hidden layer. According to the above formula, S_2 has q kinds of values:

$$S_2 = \left[S_{21}, S_{22}, \cdots, S_{2q}\right]. \tag{23}$$

Similarly, the number of hidden layer nodes in the next layer can be determined according to the number of hidden layer nodes in the second layer. Through the training results, the number of hidden layer nodes is adjusted, and the optimal value is finally determined.

Since the DBN network first trains the RBM layer and then performs error feedback training on the BP layer, the network parameters of the RBM layer and BP layer need to be set separately. Since the initial network parameter values need to be adjusted during the training process, the DBN training process is also a batch training process:

$$\text{That is, training batch} = \text{number of training samples} \div \text{number of training samples each time.} \tag{24}$$

The maximum number of cycles is also a factor that affects the DBN classification effect. This parameter also needs to be adjusted in the cycle. At the same time, the learning rate and momentum factor are also important parameters that affect the network effect.

5. Fast Pedestrian Detection Algorithm Based on Autoencoding Neural Network and AdaBoost

The framework of the autoencoding and decoding deep network is shown in Figure 6. It can be seen from the figure that the network is similar to the autoencoding network and consists of two processes: encoding and decoding.

The codec deep network structure is shown in Figure 7. It can be seen from the figure that its network structure is different from the autoencoding network. The coding process of the network is composed of multiple nonlinear mapping layers [h_1, h_2, h_3]. The three nonlinear mapping layers compress feature dimensions layer by layer to make the network's coding process remove redundant features as much as possible, so as to retain the features after the face pose changes are removed.

The convolutional autoencoder is an unsupervised algorithm model. However, convolutional autoencoders no longer use full connections in encoding and decoding, but instead use convolution operations. The convolutional autoencoder also protects the input layer, hidden layer, and output layer, and its training steps also include encoding, decoding, and parameter learning. The encoder uses a convolution operation to obtain a hidden layer from the input layer. This process is called convolutional encoding. The decoder uses a deconvolution operation to reconstruct the hidden layer to obtain an output layer with the same dimensions as the input layer, which is called convolutional decoding. Finally, the error between the output layer and the input layer is calculated, and several iterations are performed to minimize the error and obtain the convolutional autoencoder parameters. Figure 8 is a schematic diagram of the operation of the convolutional autoencoder.

The motion estimation algorithm based on block matching is simple and easy to implement. The basic principle is to find the block with the smallest matching error with the current block in the search range of the reference frame through a certain matching criterion. The relative displacement between each current block and the searched matching block is the motion vector MV, which is the motion track of the block, as shown in Figure 9.

Among the commonly used classification algorithms, the realization process of the Naive Bayes classification algorithm is relatively simple, but it cannot meet the condition of independent distribution of data in practical applications. The *K*-nearest neighbor classification algorithm is simple and effective, but the data processing is more complicated. Multilayer neural networks achieve high classification accuracy at the cost of setting a large number of initial parameters. The support vector machine algorithm (SVM) focuses on the classification of small samples and is the simplest algorithm in linear classification problems. Therefore, the SVM classifier is the most suitable for the two types of problems in this paper that only need to detect pedestrians and nonpedestrians. The flow chart of pedestrian classification is shown in Figure 10.

The target detection network is used to detect pedestrians in a certain frame of the night vision video sequence, and then, the KCF tracking model is used to track the target for several frames, forming a detection-tracking alternate mode. At the same time, by means of intermittent cyclic triggering, after the previous detection is completed for 2 seconds, the system is used to detect and track subsequent videos again, and the cycle continues until the end or termination of the video. Among them, the reason for the detection at an interval of 2 seconds is that the entire process of the target pedestrian from appearing to being caught by human eyes to leaving the field of view takes at least 2 seconds. Therefore, it is most appropriate to set it to detect once every two seconds to achieve the effect of reducing the number of detection frames, while avoiding missed detection of new targets in the field of view. The multitarget rapid detection-tracking detection cycle system is shown in Figure 11.

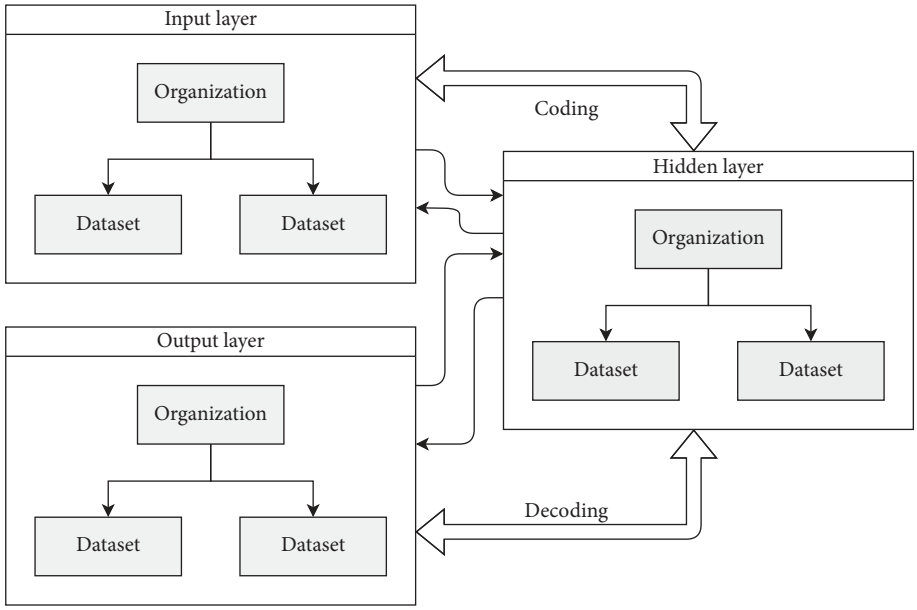

Figure 6: Block diagram of the deep network of autoencoding and decoding.

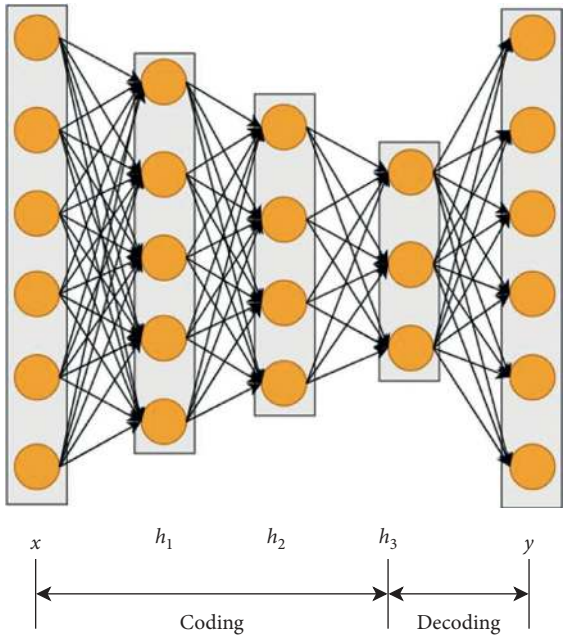

Figure 7: Autoencoding and decoding deep network.

The idea of multitarget tracking is to execute multiple tracker objects at the same time. Since this article is applied to roads where cars are driving at night and there are not many pedestrians, the tracking algorithm is still very efficient. The multitarget tracking process is shown in Figure 12.

6. Algorithm Performance Verification

Next, the performance verification of the fast pedestrian detection algorithm based on autoencoding neural network and AdaBoost constructed in this paper is carried

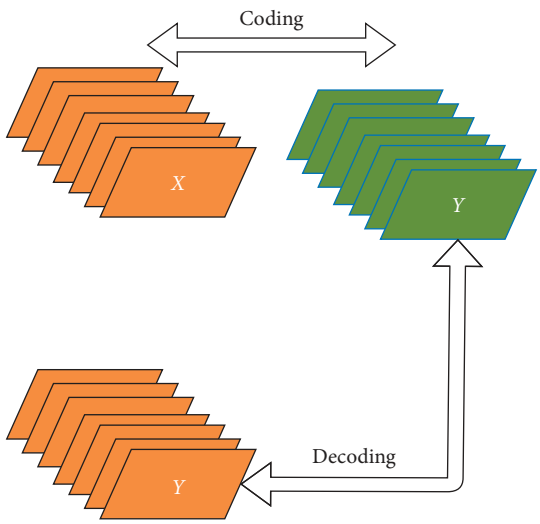

Figure 8: Convolutional autoencoder.

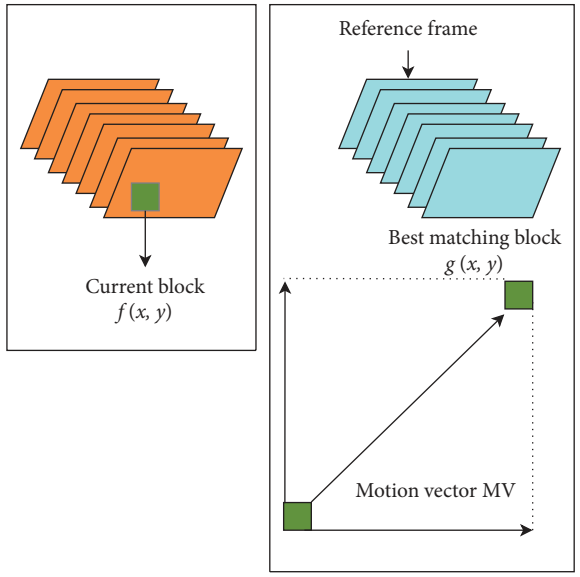

Figure 9: Schematic diagram of block matching search.

out. In order to verify the efficiency of frame selection fusion and adaptive partition fusion proposed in this paper, this paper compares it with direct fusion and evaluates it from the perspective of video fusion speed. The speed of video fusion is measured by the number of frames per second (FPS). Under the same test video, this article uses two fusion algorithms to average the time required to fuse one frame to indirectly reflect the fusion speed. In order to avoid the interference of three factors such as scene, halo area, and vehicle speed on the fusion speed, the experiment is divided into the following four groups of situations, and the influence of the third variable on the fusion speed is studied by unifying two of them. Four sets of experiments: (1) pedestrians drive at normal speed on main roads in the city, but the halo area is small; (2) pedestrians drive at normal speed on main roads in the city, but the halo area is large; (3) the halo area in the suburbs is small, but the driving speed is normal; (4) the halo area in the suburbs is small, but the driving speed is faster. The average fusion time of one frame in each group of videos recorded and the direct fusion processing speed are shown in Table 1 and Figure 13, and the selected frame-partition fusion processing speed is shown in Table 2 and Figure 14.

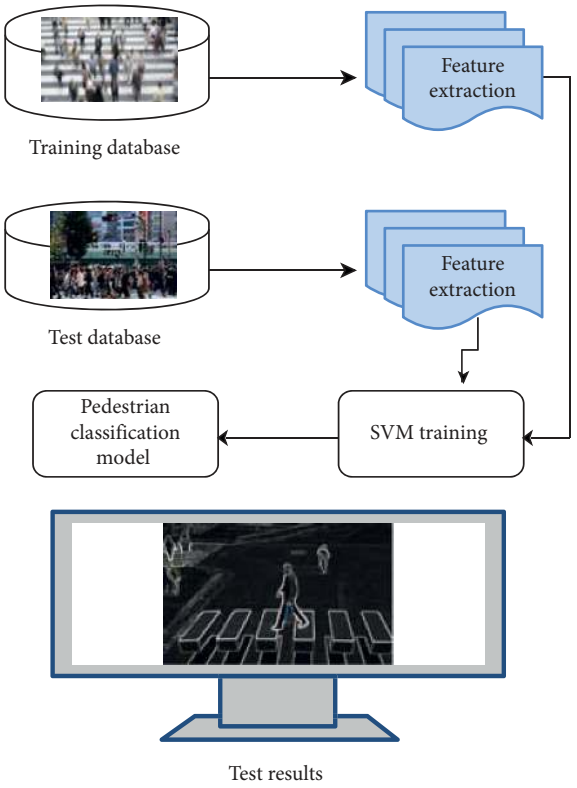

Figure 10: Pedestrian detection process.

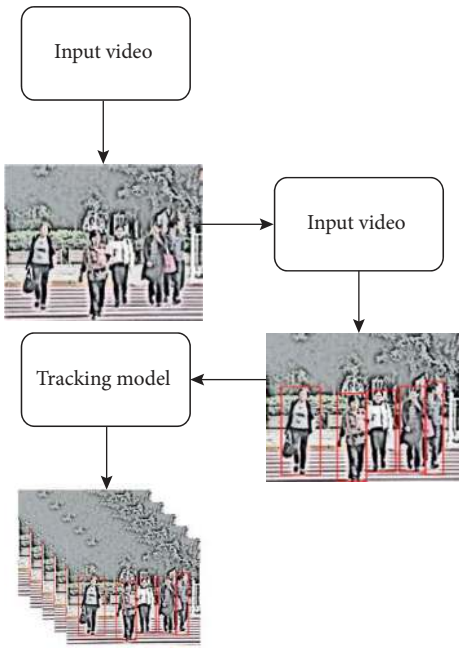

Figure 11: Pedestrian detection and multitarget tracking loop system.

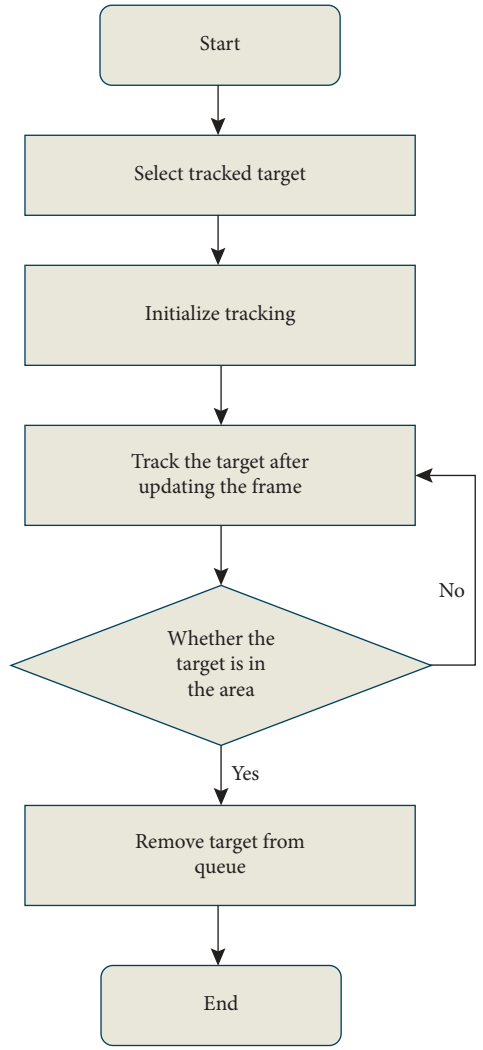

Figure 12: Multitarget tracking flowchart.

Table 1: Statistical table of processing speed in the direct fusion mode (frames/sec).

No.	Experiment 1	Experiment 2	Experiment 3	Experiment 4	No.	Experiment 1	Experiment 2	Experiment 3	Experiment 4
1	1.00	0.94	1.22	0.91	51	0.98	0.96	1.20	0.91
2	0.99	0.94	1.20	0.89	52	0.99	0.95	1.21	0.92
3	0.98	0.95	1.20	0.92	53	1.01	0.94	1.20	0.90
4	1.01	0.95	1.20	0.88	54	0.99	0.94	1.21	0.88
5	1.01	0.95	1.19	0.91	55	1.01	0.95	1.20	0.90
6	1.01	0.95	1.22	0.92	56	0.98	0.95	1.22	0.89
7	1.00	0.96	1.20	0.89	57	1.00	0.95	1.19	0.92
8	1.00	0.95	1.20	0.89	58	1.02	0.95	1.19	0.90
9	0.99	0.96	1.19	0.90	59	1.01	0.95	1.19	0.88
10	0.98	0.95	1.20	0.88	60	1.01	0.96	1.21	0.89
11	1.02	0.96	1.21	0.91	61	1.00	0.95	1.20	0.89
12	0.98	0.96	1.20	0.90	62	0.99	0.96	1.22	0.88
13	0.98	0.94	1.20	0.91	63	1.00	0.95	1.22	0.89
14	1.01	0.96	1.20	0.89	64	0.99	0.95	1.22	0.92
15	1.02	0.96	1.20	0.91	65	1.00	0.96	1.22	0.91
16	1.02	0.95	1.21	0.91	66	0.98	0.95	1.20	0.90
17	1.01	0.94	1.19	0.92	67	1.00	0.95	1.19	0.88
18	1.01	0.95	1.21	0.88	68	0.99	0.94	1.19	0.90
19	1.01	0.96	1.21	0.91	69	0.99	0.96	1.20	0.91

Table 1: Continued.

No.	Experiment 1	Experiment 2	Experiment 3	Experiment 4	No.	Experiment 1	Experiment 2	Experiment 3	Experiment 4
20	1.00	0.94	1.20	0.91	70	1.00	0.95	1.21	0.91
21	1.00	0.95	1.22	0.89	71	1.01	0.94	1.19	0.90
22	0.99	0.95	1.22	0.90	72	1.00	0.95	1.23	0.92
23	1.00	0.95	1.23	0.91	73	1.01	0.95	1.21	0.90
24	1.00	0.95	1.22	0.89	74	1.00	0.95	1.21	0.91
25	1.02	0.96	1.22	0.89	75	1.01	0.96	1.20	0.88
26	0.99	0.95	1.21	0.91	76	1.02	0.95	1.21	0.90
27	1.01	0.95	1.22	0.90	77	1.01	0.95	1.21	0.89
28	0.98	0.94	1.22	0.90	78	0.99	0.95	1.21	0.89
29	0.98	0.94	1.23	0.88	79	1.00	0.95	1.23	0.91
30	0.99	0.96	1.22	0.92	80	1.00	0.94	1.22	0.90
31	1.02	0.94	1.20	0.90	81	0.98	0.94	1.23	0.91
32	0.99	0.94	1.20	0.89	82	1.00	0.94	1.22	0.89
33	1.02	0.96	1.21	0.90	83	0.98	0.95	1.22	0.88
34	1.02	0.94	1.19	0.91	84	0.98	0.95	1.21	0.90
35	0.98	0.94	1.20	0.90	85	0.98	0.95	1.20	0.89
36	1.00	0.96	1.20	0.88	86	0.98	0.95	1.20	0.91
37	1.00	0.96	1.20	0.91	87	1.01	0.96	1.22	0.89
38	1.02	0.95	1.19	0.88	88	1.02	0.95	1.22	0.89
39	1.01	0.95	1.20	0.90	89	1.01	0.94	1.21	0.90
40	1.01	0.96	1.19	0.88	90	1.01	0.95	1.20	0.89
41	0.98	0.96	1.22	0.92	91	0.99	0.94	1.19	0.90
42	0.98	0.96	1.20	0.89	92	1.00	0.95	1.21	0.92
43	1.00	0.95	1.21	0.90	93	1.00	0.95	1.20	0.89
44	1.00	0.95	1.20	0.92	94	1.01	0.95	1.23	0.89
45	1.00	0.94	1.22	0.91	95	0.99	0.94	1.20	0.89
46	1.02	0.95	1.20	0.88	96	1.01	0.94	1.22	0.91
47	1.00	0.96	1.20	0.90	97	0.99	0.96	1.19	0.92
48	1.01	0.96	1.22	0.90	98	0.98	0.95	1.22	0.90
49	0.99	0.95	1.20	0.90	99	1.00	0.95	1.20	0.90
50	1.01	0.95	1.21	0.92	100	0.99	0.95	1.19	0.89

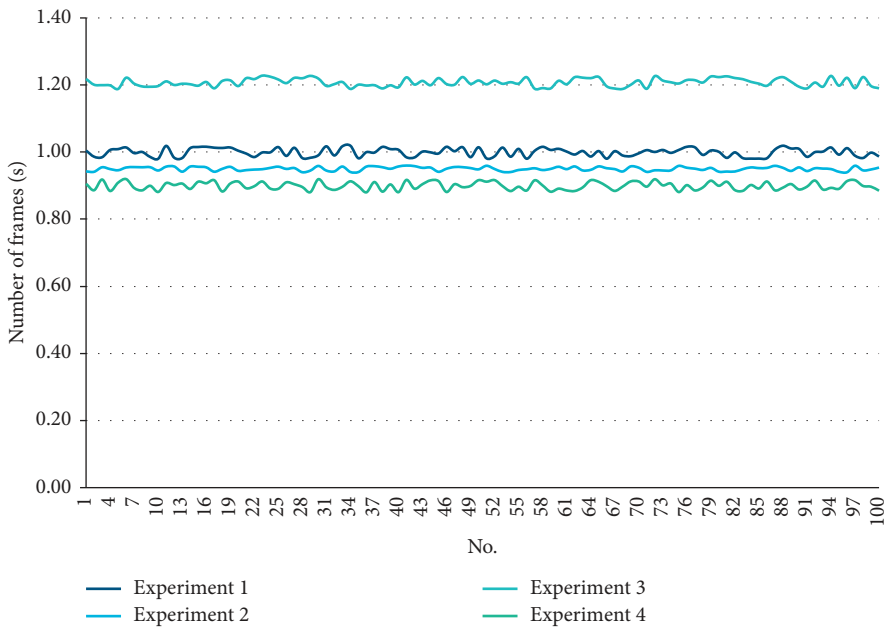

Figure 13: Statistical diagram of processing speed in the direct fusion mode (frames/sec).

Table 2: Statistical table of processing speed in the frame selection partition fusion mode (frames/sec).

No.	Experiment 1	Experiment 2	Experiment 3	Experiment 4	No.	Experiment 1	Experiment 2	Experiment 3	Experiment 4
1	6.42	5.22	6.81	5.79	51	6.38	5.34	6.71	5.87
2	6.31	5.21	6.78	5.67	52	6.33	5.20	6.87	6.00
3	6.38	5.33	6.88	5.88	53	6.53	5.32	6.88	5.78
4	6.51	5.48	6.75	5.98	54	6.65	5.36	6.78	5.61
5	6.41	5.17	6.75	5.99	55	6.39	5.41	6.81	5.90
6	6.61	5.28	6.86	5.90	56	6.60	5.12	6.78	5.82
7	6.48	5.33	6.74	5.66	57	6.42	5.26	6.82	5.86
8	6.36	5.31	6.82	5.87	58	6.35	5.18	6.74	5.95
9	6.62	5.13	6.87	5.96	59	6.65	5.18	6.87	5.98
10	6.64	5.27	6.88	5.68	60	6.69	5.21	6.84	5.80
11	6.66	5.24	6.78	5.74	61	6.66	5.39	6.81	5.83
12	6.66	5.16	6.89	5.63	62	6.51	5.26	6.80	5.70
13	6.44	5.28	6.88	5.89	63	6.69	5.40	6.89	5.87
14	6.33	5.46	6.81	5.74	64	6.50	5.18	6.79	5.65
15	6.45	5.33	6.82	5.99	65	6.55	5.25	6.77	5.60
16	6.65	5.17	6.71	6.00	66	6.67	5.45	6.89	5.98
17	6.40	5.40	6.82	5.81	67	6.66	5.51	6.82	5.83
18	6.31	5.29	6.76	5.64	68	6.48	5.24	6.81	5.95
19	6.48	5.43	6.85	5.74	69	6.51	5.49	6.84	5.91
20	6.55	5.26	6.81	5.75	70	6.48	5.31	6.81	5.86
21	6.56	5.38	6.90	5.63	71	6.46	5.32	6.82	5.98
22	6.30	5.26	6.83	5.75	72	6.52	5.48	6.90	5.91
23	6.63	5.29	6.79	5.94	73	6.37	5.28	6.70	5.81
24	6.51	5.31	6.85	5.96	74	6.49	5.38	6.85	5.67
25	6.66	5.35	6.90	5.87	75	6.54	5.13	6.71	5.61
26	6.37	5.37	6.73	5.61	76	6.68	5.15	6.74	5.94
27	6.60	5.45	6.72	5.70	77	6.56	5.29	6.73	5.87
28	6.37	5.20	6.75	5.81	78	6.34	5.37	6.84	5.97
29	6.61	5.29	6.75	5.72	79	6.65	5.20	6.80	5.78
30	6.61	5.25	6.87	5.84	80	6.53	5.35	6.75	5.61
31	6.32	5.29	6.70	5.89	81	6.37	5.28	6.85	5.90
32	6.59	5.19	6.86	5.79	82	6.62	5.34	6.76	5.72
33	6.30	5.34	6.89	5.90	83	6.32	5.26	6.79	5.66
34	6.43	5.39	6.71	5.66	84	6.62	5.38	6.71	5.96
35	6.45	5.21	6.79	5.73	85	6.45	5.47	6.73	5.97
36	6.46	5.28	6.78	5.99	86	6.66	5.12	6.70	5.86
37	6.36	5.40	6.89	5.87	87	6.45	5.23	6.84	5.89
38	6.63	5.49	6.76	5.92	88	6.34	5.24	6.76	5.87
39	6.32	5.34	6.83	5.83	89	6.31	5.19	6.72	6.00
40	6.49	5.41	6.71	5.94	90	6.51	5.41	6.71	5.62
41	6.41	5.48	6.80	5.70	91	6.46	5.38	6.82	5.71
42	6.34	5.49	6.87	5.67	92	6.37	5.34	6.76	5.72
43	6.34	5.35	6.75	5.66	93	6.46	5.13	6.74	5.68
44	6.33	5.15	6.77	5.90	94	6.37	5.38	6.79	5.80
45	6.64	5.51	6.77	5.99	95	6.44	5.38	6.88	5.88
46	6.64	5.24	6.88	5.69	96	6.35	5.19	6.73	5.90
47	6.48	5.43	6.76	5.66	97	6.66	5.48	6.77	5.88
48	6.32	5.16	6.75	5.89	98	6.54	5.26	6.76	5.84
49	6.31	5.14	6.80	5.86	99	6.51	5.16	6.76	5.93
50	6.50	5.45	6.71	5.99	100	6.67	5.34	6.75	5.99

In summary, it can be seen from the subjective and objective evaluation results of the fusion video that the algorithm proposed in this paper has the best video fusion effect, and its video is smooth, content is synchronized, and the speed of video fusion is also significantly improved.

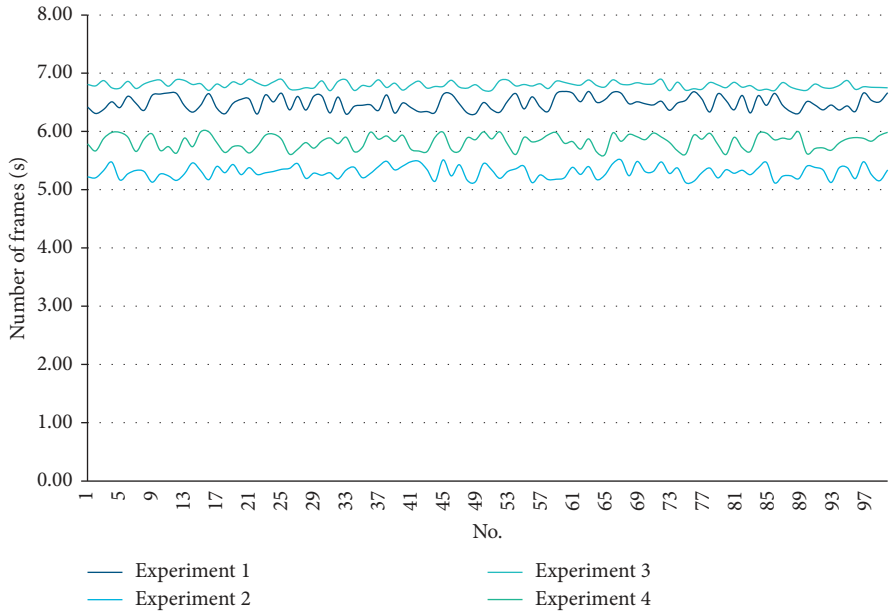

Figure 14: Statistical diagram of the processing speed in the frame selection partition fusion mode (frames/sec).

Table 3: Statistical table of the accuracy of pedestrian detection.

No.	Accuracy (%)	No.	Accuracy (%)	No.	Accuracy (%)	No.	Accuracy (%)
1	99	24	97	47	98	69	98
2	97	25	99	48	98	70	97
3	98	26	98	49	99	71	98
4	98	27	98	50	99	72	98
5	97	28	97	51	97	73	97
6	97	29	99	52	98	74	97
7	98	30	99	53	99	75	98
8	97	31	98	54	97	76	97
9	99	32	98	55	98	77	98
10	98	33	99	56	97	78	97
11	99	34	98	57	99	79	98
12	99	35	99	58	98	80	98
13	99	36	97	59	97	81	99
14	98	37	98	60	98	82	98
15	99	38	97	61	98	83	98
16	97	39	97	62	97	84	98
17	97	40	97	63	97	85	98
18	98	41	98	64	99	86	98
19	97	42	98	65	99	87	99
20	98	43	99	66	97	88	99
21	98	44	99	67	99	89	99
22	97	45	99	68	99	90	98
23	98	46	98	—	—	—	—

Next, this paper analyzes the pedestrian detection accuracy rate of the algorithm constructed in this paper. A total of 90 sets of pedestrian videos are set up. The algorithm is used to detect and count the detection results. The results are shown in Table 3 and Figure 15.

From the above results, it can be seen that the algorithm constructed in this paper has a high accuracy rate in the pedestrian detection process, so the algorithm in this paper can be applied to practice.

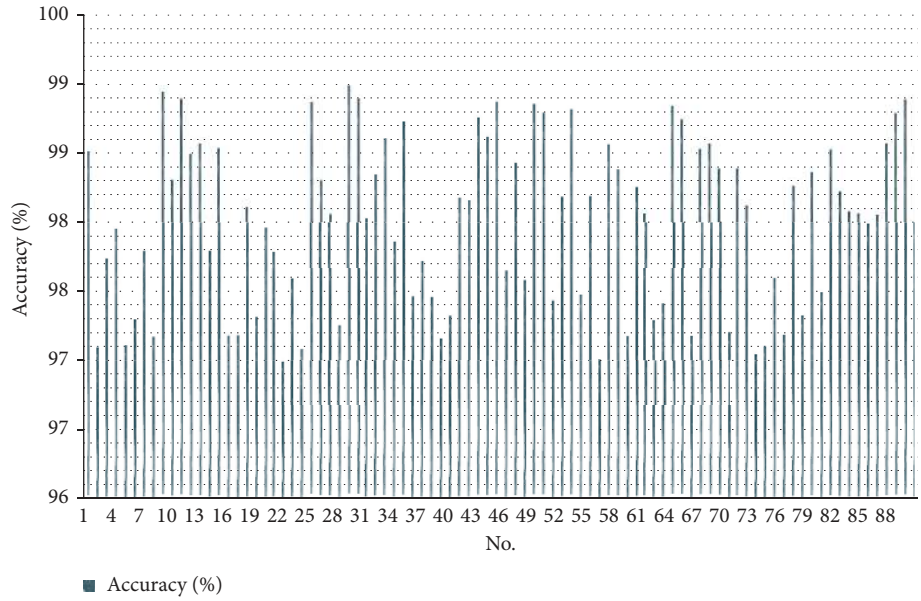

Figure 15: Statistical diagram of the accuracy of pedestrian detection.

7. Conclusion

In order to solve the problem of low pedestrian detection accuracy of real traffic cameras and high missed detection rate of small target pedestrians and improve the speed and accuracy of pedestrian detection, this paper studies pedestrian detection technology. By analyzing the classification performance of the classifier, this paper combines the AdaBoost ensemble idea with the DBN classifier and proposes a classification algorithm of autoencoding neural network and AdaBoost-DBN. Aiming at the problem that the AdaBoost algorithm is too strict for the classification accuracy of the weak classifier, this paper improves the calculation formula of the classifier weight, which effectively reduces the accuracy requirements for the weak classifier. Moreover, this paper improves the traditional AdaBoost algorithm structure, that is, resets the sample weight update formula and the strong classifier output formula and proposes a two-input AdaBoost-DBN classification algorithm. In addition, to solve the problem of unsmooth playback of the fusion video, this paper considers the motion information of the video object, performs pixel interpolation with motion compensation, and restores the frame rate of the original video by reconstructing the dropped interframe image. Finally, this paper designs experiments to verify the performance of the algorithm proposed in this paper. The results show that the results of the research meet the expected goals of algorithm construction.

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 regarding the publication of this paper.

Acknowledgments

Science Research and Innovation Team of Fuyang Normal University (kytd202004), Outstanding Talent Development Programme of College of Information Engineering, Fuyang Normal University (2018FXJT02), Anhui Provincial Education Department's Excellent Youth Talent Support Program (gxyqZD2020054), Project of Industry-University-Research Innovation Fund (2018A01010), and Anhui Provincial Education Department's Excellent Youth Talent Support Program (no. gxyq2017159).

References

[1] R. Giannetti, A. Petrella, J. Bach, A. K. Silverman, M. A. Sáenz-Nuño, and N. Pérez-Mallada, "*In vivo* bone position measurement using high-frequency ultrasound validated with 3-D optical motion capture systems: a feasibility study," *Journal of Medical & Biological Engineering*, vol. 37, no. 7, pp. 1–8, 2017.

[2] J. P. Ngel-López and N. A. De la Peńa, "Kinematic hand analysis using motion capture technology," *IFMBE Proceedings*, vol. 49, no. 6, pp. 257–260, 2015.

[3] M. A. Khan, "Multiresolution coding of motion capture data for real-time multimedia applications," *Multimedia Tools & Applications*, vol. 76, no. 15, pp. 16683–16698, 2017.

[4] O. M. Khudolii, O. V. Ivashchenko, S. S. Iermakov, and O. G. Rumba, "Computer simulation of junior gymnasts' training process," *Science of Gymnastics Journal*, vol. 8, no. 3, pp. 215–228, 2016.

[5] M.-K. Kim, T. Y. Kim, and J. Lyou, "Performance improvement of an AHRS for motion capture," *Journal of Institute of Control, Robotics and Systems*, vol. 21, no. 12, pp. 1167–1172, 2015.

[6] Y. Lee and H. Yoo, "Low-cost 3D motion capture system using passive optical markers and monocular vision," *Optik International Journal for Light & Electron Optics*, vol. 130, no. 2, pp. 1397–1407, 2017.

[7] T. Miura, T. Kaiga, T. Shibata, K. Tajima, and H. Tamamoto, "Low-dimensional feature vector extraction from motion

capture data by phase plane analysis," *Journal of Information Processing*, vol. 25, no. 6, pp. 884–887, 2017.

[8] D. Mulligan, K. R. Lohse, and N. J. Hodges, "An action-incongruent secondary task modulates prediction accuracy in experienced performers: evidence for motor simulation," *Psychological Research*, vol. 80, no. 4, pp. 496–509, 2016.

[9] S. W. Park, H. S. Park, J. H. Kim, and H. Adeli, "3D displacement measurement model for health monitoring of structures using a motion capture system," *Measurement*, vol. 59, no. 5, pp. 352–362, 2015.

[10] A. Puupponen, T. Wainio, B. Burger, and T. Jantunen, "Head movements in Finnish sign language on the basis of motion capture data," *Sign Language and Linguistics*, vol. 18, no. 1, pp. 41–89, 2015.

[11] M. M. Rahman, "Analysis of finger movements of a pianist using magnetic motion capture system with six dimensional position sensors," *Transactions of the Virtual Reality Society of Japan*, vol. 15, no. 3, pp. 243–250, 2017.

[12] K. R. Ridderinkhof and M. Brass, "How kinesthetic motor imagery works: a predictive-processing theory of visualization in sports and motor expertise," *Journal of Physiology-Paris*, vol. 109, no. 1-3, pp. 53–63, 2015.

[13] S. K. Wang, H. Xie, and B. Hu, "Research on protection property of running sportswear fabrics based on 3-D motion capture system," *Textiles and Light Industrial Science and Technology*, vol. 3, no. 2, pp. 57–62, 2014.

[14] H. Zhang, L. Wang, S. Chu, S. Chen, H. Meng, and G. Liu, "Application of optical motion capture technology in power safety entitative simulation training system," *Optics and Photonics Journal*, vol. 6, no. 8, pp. 155–163, 2016.

[15] Z.-M. Zhou and Z.-W. Chen, "A survey of motion capture data earning as high dimensional time series," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 10, no. 9, pp. 17–30, 2015.

[16] K. A. Tivener and D. S. Gloe, "The effect of high-fidelity cardiopulmonary resuscitation (CPR) simulation on athletic training student knowledge, confidence, emotions, and experiences," *Athletic Training Education Journal*, vol. 10, no. 2, pp. 103–112, 2015.

[17] O. V. Ivashchenko and O. O. Kapkan, "Simulation of process of 14-15 years old girls' training of light athletic and gymnastic exercises," *Pedagogics, Psychology, Medical-Biological Problems of Physical Training and Sports*, vol. 19, no. 8, pp. 32–39, 2015.

[18] P. D. Owen and N. King, "Competitive balance measures in sports leagues: the effects of variation in season length," *Economic Inquiry*, vol. 53, no. 1, pp. 731–744, 2015.

[19] J. Yang, "The simulation of table tennis during the course of sports," *Caribbean Journal of Science*, vol. 52, no. 4, pp. 1561–1564, 2019.

[20] M. Bulat, N. Korkmaz Can, Y. Z. Arslan, and W. Herzog, "Musculoskeletal simulation tools for understanding mechanisms of lower-limb sports injuries," *Current Sports Medicine Reports*, vol. 18, no. 6, pp. 210–216, 2019.

[21] E. C.-H. Lin, "A research on 3D motion database management and query system based on kinect," *Lecture Notes in Electrical Engineering*, vol. 329, no. 1, pp. 29–35, 2015.

[22] H. Shan and Y. Liu, "Feature recognition of body dance motion in sports dancing," *Metallurgical and Mining Industry*, vol. 7, no. 7, pp. 290–297, 2015.