Similar Gesture Recognition via an Optimized Convolutional Neural Network and Adam Optimizer

Ya Gao, Chenchong Jia, Yifei Qiao, Xi Huang, Juan Lei, and Xianwei Jiang(✉)

School of Mathematics and Information Science, Nanjing Normal University of Special Education, Nanjing 210038, China
jxw@njts.edu.cn

Abstract. The recognition significance of similar sign language (or confusing gesture) in sign language recognition is highlighted, and the goal is to realize the recognition of such gesture and sign language based on deep learning with an optimized convolutional neural network and the Adam optimizer. The convolutional layer and the pooling layer are connected alternately. The locally connected image data and parameter features are used to extract the shared pooling layer, and the image resolution reduction of image data sampling and the reducibility of iterative training are used to achieve the extraction precision requirements of feature points. In addition, the information transfer between layers is realized through convolution, the introduction of pooling layer and RELU activation function to realize nonlinear mapping and reduce the data dimension. We also use the batch normalization method for faster convergence and dropout method to reduce overfitting. Ten experiments were carried out on a nine-layer "CNN-BN-ReLU-AP-DO" method, with an average accuracy of $97.50 \pm 1.65\%$. The overall accuracy is relatively high, and gesture recognition can be conducted effectively.

Keywords: Gesture recognition · CNN · Batch normalization · Dropout · Adam · Data augmentation

1 Introduction

As the development of artificial intelligence, human-computer interaction is becoming more and more popular. The main media to study human-computer interaction are different Convolutional Neural networks (CNN) or different machine learning methods. As a natural way of communication between people, gesture has gradually become a new way of human-computer interaction. More and more people are joining the field of human-computer interaction. In the field of human-computer interaction, gestures are usually used to convey some simple instructions. Using gestures to operate a computer is usually imperative, but this is not what people use gestures for. Gesture can be divided into dynamic gesture and static gesture. As a transient state of dynamic gesture, static

W. Fu et al. (Eds.): ICMTEL 2021, LNICST 388, pp. 43–61, 2021.
https://doi.org/10.1007/978-3-030-82565-2_4

gesture is of great significance for understanding dynamic gesture. In the research process of gesture recognition, the traditional method is to segment gestures from pictures and classify them under different rules and algorithms. However, the hand is a non-rigid object with abundant changes. The changes of different environments and spaces and the deviations of gestures of individual movement make gesture recognition more difficult.

At present, gesture recognition has been widely studied, including based on neural network, vision, variable target classification task, Kinect gesture recognition technology and so on. JongShill Lee, et al. from Inda University in South Korea and Korea Polytechnic University used entropy analysis to segment the gesture region from the background complex video stream and conduct gesture recognition [1]. They detected the contour of the gesture region with chain code method, and finally calculated the distance from the center of mass of the gesture region to the contour boundary. The system can recognize 6 gestures, with an average recognition rate of over 95%, and the recognition rate of 6 individuals for each gesture reaches an average of 90%–100% [2]. Beijing Jiaotong University Li Jiang, et al. on the basis of data glove, analysis the geometric relationships between a hand shape, set up the model of virtual hand, by the data glove data interface to get the knuckles of the flex point of view, establish gestures standard sample library, and puts forward the gesture recognition method based on BP neural network, with gesture training standard sample, make it have the function that identify gestures, and by using VC+ + programming to realize BP neural network, using Matlab to prove the validity of the results (see Fig. 1) [3].

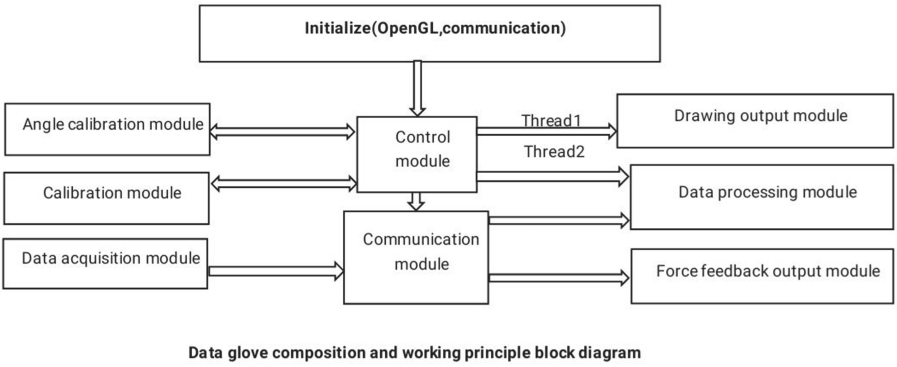

Fig. 1. The structure and work principle of data glove

There are three methods for gesture recognition: template matching, neural network and hidden Markov model [4]. A gesture is a motion of the body that contains information. Neural network gesture recognition requires a certain learning stage, and in the process of processing, there may be serious shortcomings such as large number of middle layer neurons, too long learning time, and too wide range of combining coefficients. The gesture recognition based on vision uses camera to collect the sequence of gesture image, and recognizes the gesture through image processing and analysis. This method is simple to input, low to the equipment requirements, but the recognition rate is low.

Variable target based gesture recognition can improve the efficiency of gesture recognition without increasing the amount of computation, but it has many parameters and a large amount of computation. Kinect-based gesture recognition technology is highly dependent on machine learning, which makes it easier to recognize new gestures, but more difficult to establish the system.

However, in the process of sign language recognition, we found that some single samples have low ROC values. For instance, some ROC values are between 0.76 and 0.82. Meanwhile, based on actual experience, these gestures are easy to confuse due to their similarities, such as Y and T, N and M, SH and NG, etc. Therefore, similar gesture recognition is proposed as research content, and related experiments and tests have been carried out.

In this paper, a gesture recognition algorithm using convolutional neural network is proposed, which avoids complex preprocessing of gestures and can directly input the original gesture images. With the characteristics of local perception region, hierarchical structure, feature extraction and classification process, convolutional neural network has been widely applied in the field of image recognition. Moreover, it has high accuracy, low complexity, good robustness, and overcomes many inherent shortcomings of traditional algorithms.

2 Dataset

In order to analyze the different and similar sign language and gesture pairs scientifically, we need to analyze the sign language images. To this end, we quoted the relevant sign language self-built data set [5], and used image processing techniques to highlight gesture features, thereby making our data richer, more accurate, and more comprehensive. At the same time, the use of image processing technology reduces the uncertainty and contingency of experimental data. In order to complete this research, it is necessary to process the data in the data set more carefully, putting together images that represent the same meaning (see Fig. 2).

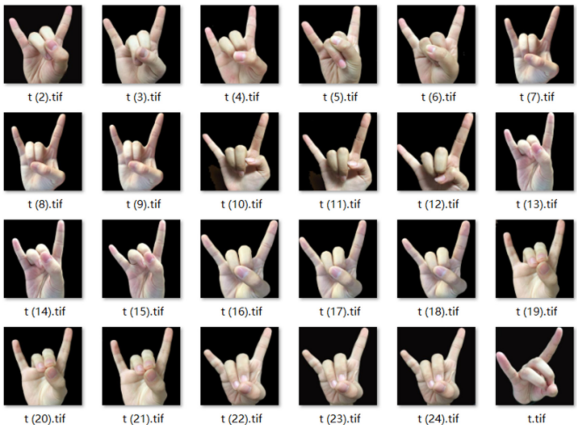

Fig. 2. The images representing the same pronunciation are put together

The data set sample used this time are the most similar and most confusing samples selected from 26 basic letters and 4 words with the pronunciation of rolling voice, and a total of six categories. The image size is 256 × 256. At the same time, the background color is uniformly set to RGB (0, 0, 0) to facilitate the realization of the training of the convolutional neural network in the following experiments.

Due to the high degree of similarity and small difference between different sign language pictures, in order to improve the accuracy of the experiment, in the stage of data screening feature samples, the following principles need to be followed [6]:

1. The sample selected from the data set needs to represent the characteristic hand shape of a specific sign language word, which is a basic requirement;
2. The samples selected from the data set are as different as possible from each other;
3. The selected samples from the data set need to include the past situation from different perspectives to reduce the influence of external factors;

Based on the above sample selection principle, in the existing sign language data set, we created a set of smaller data sets for analysis based on the sign language hand type with the smaller Roc value, which included samples of NG, SH, M, N, Y and T (see Fig. 3).

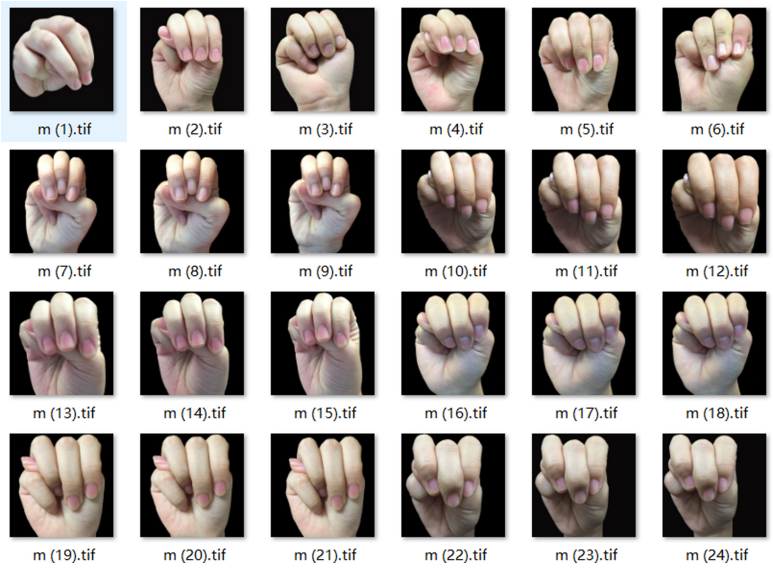

Fig. 3. Part of sample M

3 Methodology

3.1 Convolutional Layer

The convolutional layer and the pooling layer are alternately connected. The convolutional layer makes use of the local connectivity and parameter sharing of image data for feature extraction, while the pooling layer makes use of the reducibility of sampling of image data for image resolution reduction, and repeated iterative training achieves the precision requirements of feature point extraction.

The convolutional layer is the core layer of the convolutional neural network. In the multi-layer neural network architecture, the input of neurons in the latter layer is extracted from the important feature points in the local image area of the former layer, and the full connection layer in the latter performs combination matching of features, and performs classification or other tasks [7].

Convolutional layer is a process in which several feature maps are obtained through convolution calculation of data through filter to extract features from input data. Convolutional layer by using network and local connection and convolution kernel parameters share the way connected with the previous layer, a matrix that input data is local element contact more closely, distant element correlation is weak, so each neuron need only perceive the local area of the input data, perception of each local area will be connected to the next layer, this is the local connection of convolution layer; Convolution kernel sharing means that every pixel value on the characteristic surface shares the convolution kernel, which can reduce the network parameters and reduce the risk of overfitting [8]. The convolutional layer mainly has two key operations: local association and window sliding. Local association is to place each neuron as a convolution kernel; Window sliding is to set the size of filter, quantity, step length of window sliding and filling value of changing the size of feature map to carry out convolution operation on local data. The convolution operation is shown in Fig. 4. Parameter sharing is realized in all neurons of feature map, which greatly reduces the number of weights, improves the learning efficiency, and is very helpful for network training [9]. Suppose that a given two-dimensional image I (a,b), called "input", convolved with the kernel function K (p,q) is

$$S(a, b) = (I*K)(a, b) = \sum_{p} \sum_{q} I(p, q)K(a - p, b - q) \tag{1}$$

Where (p, q) represents the size of kernel. As the convolution is commutative, the formula is equivalent to

$$S(a, b) = (I*K)(a, b) = \sum_{p} \sum_{q} I(a - p, b - q)K(p, q) \tag{2}$$

3.2 Pooling Layer

The deep neural network realizes the information transfer between layers mainly through convolution, and introduces pooling layer to reduce the data dimension. The activation function realizes the nonlinear mapping. Pooling layer is equivalent to a filter, which

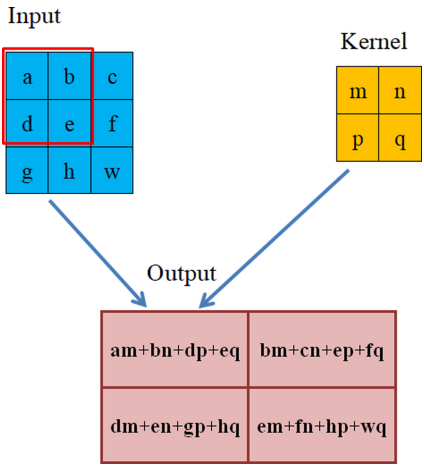

Fig. 4. Illustration of the convolution operation

is used to remove a lot of redundant information in the feature map obtained by convolutional layer and activation layer and to screen out the most representative feature information. The function of pooling layer is to gradually reduce the spatial size of the data body, so as to reduce the number of parameters in the network, reduce the consumption of computing resources, and effectively control the overfitting. The pooling layer mainly contains two parameters, namely the step size and the size of the pooling core. The pooling kernel processes the input feature graph in the way of sliding window, and the corresponding key features are obtained through the calculation of different pooling functions. Different pooling functions mean different pooling methods.

Maximum pooling core average pooling is the most common pooling method. The maximum pooling layer only retains the maximum value in the pooling box, so the most representative information in the feature diagram can be extracted effectively [10]. Averaging pooling can calculate the mean of all the values in the pooling box, so that all the information in the feature graph can be averaged, without losing too much key information. Figure 5 shows an example of a maximum pool and an average pool.

Given the pool area R, the activation set M contained in R is

$$M = \{m_i | i \in R\} \tag{3}$$

Then the max pooling P_m can be expressed as

$$P_m = \max(M_R) \tag{4}$$

Another pool strategy, the average pooling P_a is defined as:

$$P_a = \frac{\sum M_R}{|M_R|} \tag{5}$$

Where M_R is the number of elements in the set M.

Fig. 5. Examples of max pooling and average pooling

Pooling helps the input representation to be approximately constant and reduces the computational burden. But both the max pooling and the average pooling have their own disadvantages. The former is easy to over fit training data and can only reduce the estimated average offset due to convolutional layer parameter errors and cannot be generalized to the test set. The latter may reduce the intense activation when many elements in the pool area are close to zero, and it only reduces the error estimated variance due to the size of the finite field.

In order to overcome the shortcomings of the maximum pool method and the average pool method, we introduce the Stochastic Pooling (SP). Stochastic Pooling replaces the traditional deterministic pooling operation with a random process, which is activated by randomly selecting a value in each pooling region according to the polynomial distribution given by the activity of the pooling region. To be more specific, firstly, the probability p of each region is calculated through the activation within the planning region. The specific formula is as follows:

$$p_i = \frac{b_i}{\sum b_k}, \quad k \in R_j \tag{6}$$

Where R_j is the pooling area, it is easy to sees that the greater the element value in each pooling domain is, the greater the probability P is. Then, a value in the pooling domain is randomly selected as the final value according to the probability value. Due to the randomness of Stochastic Pooling, it retains the advantages of max pooling and average pooling to a large extent, and can avoid the fitting of convolutional neural network [11].

3.3 ReLU

ReLU is an activation function commonly used in artificial neural networks, and it is a linear correction function. It is defined as:

$$f(x) = \begin{cases} x \text{ if } x > 0, \\ 0 \text{ if } x \leq 0, \end{cases} \tag{7}$$

If the input value is less than or equal to 0, then the output value is 0, otherwise the original value remains unchanged. This is a method to force some data to be zero, but it has been proved by practice that the trained network is completely moderately sparse. Thus, the interdependent relationship between parameters is reduced and the occurrence of overfitting is alleviated [12]. ReLU function effectively solves the problem of gradient disappearance of sigmoid series functions, but it still has the disadvantage of mean deviation. When $x < 0$, it is hard saturated [13]. If the input results fall into this region, the gradient of the neuron will always be 0 and will not activate any data again, that is, the neuron will die, which directly leads to the non-convergence of the calculation results. And the output of the ReLU function is 0 when $x < 0$, making the overall output mean greater than 0, which cannot alleviate the mean deviation problem.

Leaky ReLU (LReLU) is a variant of ReLU. Its proposal is to solve the Dead ReLU problem, where ReLU sets all negative values to zero and LReLU gives all negative values a non-zero slope. It is defined as:

$$y_i = \begin{cases} x_i \text{ if } x_i \geq 0, \\ \frac{x_i}{\alpha_i} \text{ if } x_i < 0, \end{cases} \tag{8}$$

Where α_i is a fixed parameter in $(1, +\infty)$. However, in some experiments, we found that LReLU had no significant influence on accuracy.

Parametric ReLU (PReLU) function is a modified version of LReLU function, which is unsaturated and can alleviate mean shift and neuronal death, and is defined as follows:

$$f(x) = \begin{cases} x \text{ if } x > 0, \\ \alpha x \text{ if } x \leq 0, \end{cases} \tag{9}$$

Compared with ReLU function, when x is less than or equal to 0, the slope coefficient α of PReLU function can be determined according to the data, or it can learn parameters from the data adaptively, instead of being fixed, and the output mean approaches 0. Moreover, the function is not hard saturated at this time, so the convergence speed of PReLU function is faster and there is no problem of neuron death.PReLU can be used for back propagation training and can be optimized in parallel with other layers [14].

Randomized Leaky ReLU (RReLU) is another improvement on LReLU. In RReLU, the negative slope is random in training and fixed in later tests. The highlight of RReLU is that α_{ji} is a value randomly selected from a uniform distribution U(I, U) during the training. Formally, we can get the following results:

$$y_{ji} = \begin{cases} x_{ji} \text{ if } x_{ji} \geq 0, \\ \alpha_{ji} x_{ji} \text{ if } x_{ji} < 0, \end{cases} \tag{10}$$

Where $\alpha_{ji} \sim U(l, u)$, $l < u$ and $l, u \in [0, 1)$.

The core idea of RReLU is that, during training, α is a random value from a Gaussian distribution U (l,u), which is then modified during testing. In the testing stage, take the average value of α_{ji} during the training process. PReLu is a bit of a regularization.

3.4 Batch Normalization

Batch Normalization is a very simple but easy-to-use practical technique that can reduce the problem of internal covariate shift and speed up training convergence. For deep neural networks, even if the output data of the previous layer has been "whitened", the update of various model parameters during training will still easily cause the hidden layer input distribution of the corresponding layer in the network to shift or Changes cause instability of calculated values. This instability due to drift and change usually makes it difficult for us to train an effective depth model. Batch Normalization adjusts the activation value of a number of training examples by selecting several min-batch data sets in each selected training data set. After activation, the input value of any neuron in each layer of the neural network will be forced back to the standard normal distribution with a mean of 0 and a variance of 1, which effectively reduces the influence of the instability of the calculated value.

In order to improve the accuracy of network training to a certain extent and prevent the decrease of network expression ability during network training, the selection of min-batch data set in Batch Normalization training and inference should consider the distribution ratio of all training data, so that the neural network The prediction effect will be better, and the training accuracy of the network will be improved to a certain extent. After adopting Batch Normalization, the model parameter gradient will be less affected by the changes of various model parameters during the training update process, which speeds up the convergence speed [15].

3.5 Dropout

Dropout is a commonly used method to suppress over-fitting in deep learning. The method is to randomly delete a part of neurons during the neural network learning process. During training, a part of neurons are randomly selected and their output is set to 0. These neurons will not transmit signals to the outside [16]. Figure ab is a schematic diagram of Dropout, on the left is the complete neural network, and on the right is the network structure after applying Dropout. After applying Dropout, the marked neurons will be deleted from the network so that they will not transmit signals to the subsequent layers. Randomly identified neurons are discarded in the learning process, so the model does not rely too much on certain neurons, which can inhibit overfitting to a certain extent.

Dropout temporarily discards a part of neurons and their connections. Randomly discarding neurons can prevent overfitting, while connecting different network architectures exponentially and efficiently. The probability of neurons being discarded is $1 - p$, which reduces co-adaptation between neurons. The hidden layer usually discards neurons with a probability of 0.5. Use the complete network (the output weight of each node is p) to approximate the sample average of all 2^n dropout neurons. Dropout significantly

reduces overfitting, and at the same time improves the learning speed of the algorithm by avoiding training nodes on the training data.

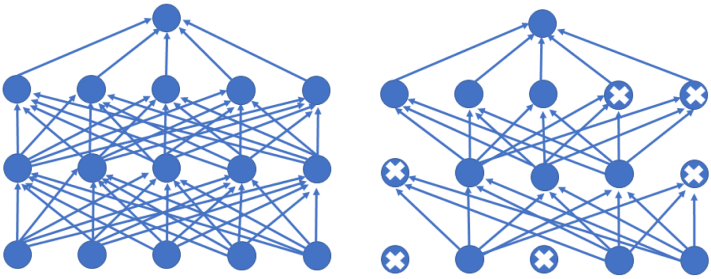

Fig. 6. Ordinary neural networks (left). Neural network with dropout (right)

When predicting the scene, the Dropout diagram in Fig. 6 will forward all neuron signals, which may lead to a new problem: because some neurons are randomly discarded during training, the total size of the output data will be reduced. For example: the calculation of its norm will be smaller than when Dropout is not used, but the neurons are not discarded during prediction, which will result in a different distribution of data during training and prediction. There are two ways to solve this problem:

1. During training, a part of neurons are randomly discarded in proportion, and their signals are not transmitted backward; during prediction, the signals of all neurons are transmitted backward, but the value on each neuron is multiplied by.
2. A part of neurons are randomly discarded in proportion during training, and their signals are not transmitted backward, but the values on those neurons that are reserved are divided by the signals of all neurons are transmitted backward during prediction, without any processing.

3.6 Data Augmentation

In order to better analyze sign language images, that is, to more accurately identify and distinguish similar sign languages, we validated some DA techniques, such as PCA color augmentation, affine transform, noise injection, scaling, random shift and gamma correction. Each part is explained in detail:

PCA Color Enhancement: First calculate the mean and standard deviation according to the three color channels of RGB, and then calculate the covariance matrix on the entire training set, perform eigen decomposition, and obtain eigenvectors and eigenvalues for PCA Data augmentation [17];

Affine Transformation: Also known as affine mapping, it means that in geometry, a vector space undergoes a linear transformation and is connected to a translation to transform it into another vector space [18]. Affine transformation can maintain the "flatness" of the image, including rotation, scaling, translation, and miscutting operations. Generally speaking, the affine transformation matrix is a matrix of 23, the elements in the third

column play the role of translation, the numbers in the first two columns are scaled on the diagonal, and the rest are rotation or crosscutting. Affine transformation is a linear transformation from two-dimensional coordinates (x,y) to two-dimensional coordinates (α,β). The mathematical expression is as follows:

$$\begin{cases} \alpha = a_1 x + b_1 y + c_1 \\ \beta = a_2 + b_2 y + c_2 \end{cases} \tag{11}$$

The corresponding homogeneous coordinate system is as follows:

$$\begin{bmatrix} \alpha \\ \beta \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{12}$$

Affine transformation maintains the "flatness" (straight lines are still straight lines after affine transformation) and "parallelism" (the relative positional relationship between straight lines remains unchanged, and parallel lines remain unchanged after affine transformation. It is a parallel line, and the position order of the points on the line will not change). Three pairs of non-collinear corresponding points determine a unique affine transformation.

Noise Injection: There are many kinds of noise injection in the neural network, such as input layer, hidden layer, weight, output layer, etc. Here only the input layer x and output layer y are used as examples.

In fact, injecting noise into the input layer can actually be regarded as a means of data set enhancement, which is essentially regularization. The reason is that the neural network is not robust to noise, so it has to be mixed with noise and then trained to improve robustness [19].

Injecting noise into the output layer is actually modeling label noise. The label of most data sets always has a certain error rate. For example, assume that the error rate E, as the number of classes k, this time is smoothed to 0, 1 is a flat slide to $1 - E$, the positive integer of the k-th output softmax.

Zoom: The image can be zoomed outward or inward. When zooming out, the final image size will be larger than the original image size. Most image frames cut out a part from the new image whose size is equal to the original image. We will deal with inward scaling in the next section because it reduces the size of the image and forces us to make assumptions about content that is beyond the boundary.

Gamma Correction: In the process of image formation, there are many factors that affect the clarity of the image. Gamma correction can help us improve the quality of the output image, so that we can distinguish more accurately when recognizing similar sign languages. The following will explain the reasons why gamma correction can improve the quality of the output image [20]. At the same time, we will also explain the importance of image quality in recognizing sign language.

Gamma correction is ignored in many graphics books. The results of lighting and shading calculations are directly output to the screen without modification, so the resulting image may not be what we expect. In fact, computer monitors (CRT and now LCD

monitors) react nonlinearly to pixel brightness. For example, without using gamma correction, a gray value of 0.5 is not half of the brightness of 1.0, and its brightness is darker than half of 1.0. The nonlinear relationship of display brightness can usually be fitted using an exponential function. Perceived brightness (P) is proportional to the index of pixel brightness (l), and this index is called gamma r. For display devices, usually the value of r is between 2.0 and 2.4. This value can be accurately determined through display calibration. We can correct this problem through gamma correction. Gamma correction compensates for the non-linear response of the display to the brightness value by performing an exponential operation on the brightness value of the pixel, so that the display result looks linear. When rendering, we first ignore the problem that the brightness response of the display is non-linear. Before the result is written into the frame buffer object, we use gamma correction to correct this problem, thereby improving image quality.

As we all know, the quality of a picture will greatly affect the human visual experience. Similarly, in the process of sign language recognition, the quality of the picture will also have a great impact on the result of sign language recognition. In particular, for some similar pictures, different test results are likely to be produced due to subtle differences in picture quality. So here we introduce Gamma correction to improve the accuracy of sign language recognition by improving the quality of the picture.

3.7 Train Algorithms

ADAM (The Adaptive Momentum) Algorithm: Set a global learning rate ρ(recommend default $\rho = 0.001$), moment estimated exponential decay rate ${}^{c}\beta_1$ and ${}^{c}\beta_2$ (${}^{c}\beta_1$ and ${}^{c}\beta_2$ in the interval [0,1) the recommended default were 0.9 and 0.990), initialization parameters for K, a small constant created for numerical stability (recommended) by default $K - 1$), the first and second moment variables s and r with an initial value of 0, and a time step counter t (initialized $t = 0$). Then there is the main body of the algorithm, which loops through the following steps and will not stop until the conditions for stopping are reached.

(1) Take out the small batch of data $\{X_1, X_2, X_3, X_4 \ldots\ldots X_m\}$ The target corresponding to the data is t represented by y_i.
(2) Calculate the gradient based on the small batch data according to the following formula:

$$g \leftarrow \frac{1}{m} \vee_{\omega} \sum_{i} L\big(f(x_i; \omega), y_i\big) \tag{13}$$

(3) Refresh time step:

$$t \leftarrow t + 1 \tag{14}$$

(4) Update the first-order partial moment estimation:

$$s \leftarrow \rho_1 s + (1 - \rho_1) g \tag{15}$$

(5) Update the second-order partial moment estimation:

$$r \leftarrow \rho_2 r + (1 - \rho_2) g \odot g \tag{16}$$

(6) Correct the deviation of the first-order moment:

$$\tilde{s} \leftarrow \frac{s}{1 - \rho_1^t} \tag{17}$$

(7) Correct the deviation of the second moment:

$$\tilde{r} \leftarrow \frac{s}{1 - \rho_2^t} \tag{18}$$

(8) Calculate the update amount of parameters:

$$\Delta\omega = -\sigma \frac{\tilde{s}}{\sqrt{\tilde{r} + \delta}} \tag{19}$$

(9) According to the $\Delta\omega$ updated parameters:

$$\omega \leftarrow \omega + \Delta\omega \tag{20}$$

In Adam, momentum is directly incorporated into the estimation of the first moment of the gradient (exponentially weighted). The most intuitive way to add momentum to RMS Prop is to apply momentum to the scaled gradient. There is no clear theoretical motivation for the use of momentum combined with scaling. Secondly, Adam includes offset correction, which corrects the estimation of the first moment (momentum term) and (non-central) second moment initialized from the origin.

Stochastic Gradient Descent With Momentum (SGDM): The momentum gradient descent method is an improved version of the gradient descent method. Generally speaking, the optimization effect is better than the gradient descent method. The core of the momentum gradient descent method is the exponentially weighted average of a series of gradients.

For example, the gradient sequence obtained in 100 gradient descent is

$$\left\{ \nabla W_1, \nabla W_2 ..... \nabla W_{99}, \nabla W_{100} \right\}$$

Then the momentum gradients corresponding to them are respectively

$$\begin{aligned} V_{\nabla W_0} &= 0 \\ V_{\nabla W_1} &= \beta V_{\nabla W_0} + (1-\beta)\nabla W_1 \\ V_{\nabla W_2} &= \beta V_{\nabla W_1} + (1-\beta)\nabla W_2 \\ &\cdot \\ &\cdot \\ &\cdot \\ V_{\nabla W_{100}} &= \beta V_{\nabla W_{99}} + (1-\beta)\nabla W_{100} \end{aligned} \tag{21}$$

The gradient after exponential weighted average is used to replace the original gradient for parameter update, because each gradient after exponential weighted average contains the information of the previous gradient.

RMS Prop: RMS prop is similar to first update vector of Adadelta,

$$E\left[g^2\right]_{i+1} = 0.9E\left[g^2\right]_i - 0.1g_i^2 \tag{22}$$

The update rules of RMS prop are as follows:

$$\theta_{i+1} = \theta_i - \frac{\alpha}{\sqrt{E\left[g^2\right]_i + \in}} \frac{\partial}{\partial \theta_i} L(\theta_i : (y_i, \hat{y}_i) \tag{23}$$

In RMS prop, the learning rate is divided by the exponential decay average of the squared gradient.

Stochastic gradient descent updates the parameters of each training sample, and updates each time, and the execution speed is faster. But the problem is that due to frequent updates and fluctuations, it will eventually converge to a minimum and there will be frequent overshoot due to fluctuations. Dam algorithm can calculate the adaptive learning rate of each parameter. This method not only stores the exponential decay average of AdaDelta's previous squared gradient, but also maintains the exponential decay average of the previous gradient M(t), which is similar to momentum. In practical application, Adam method is more effective than other adaptive learning rate algorithms, and can correct the problems existing in other optimization techniques.

4 Experiment Results

4.1 Structure of Proposed CNN

A customized 9-layer CNN model was built, which included seven convolutional layer blocks and two fully connected layer blocks. In each convolutional layer block, advanced techniques such as batch normalization, ReLU and average pooling were employed. The details of hyperparameters were listed in Table 1. Meanwhile, the value of pooling size was set as 3×3, stride and padding were defined as 2 and 1, respectively. All settings were based on fine-tuned method. Dropout technique was applied before first fully connected layer (FCL1). The dropout rate was set to 0.4. Furthermore, softmax function was adopted after second fully connected layer (FCL2). By this way, the size of output is shrunk layer by layer. At the same time, the features of each layer are automatically extracted.

Table 1. Conv layers' hyperparameters

Index	Block	Filter size	Filter number	Stride	Padding
1	Conv1_BN_ReLU	3	8	2	1
2	Conv2_BN_ReLU_Pool	3	8	2	1
3	Conv3_BN_ReLU	3	16	1	1
4	Conv4_BN_ReLU_Pool	3	16	1	1
5	Conv5_BN_ReLU	3	32	1	1
6	Conv6_BN_ReLU_Pool	3	32	1	1
7	Conv7_BN_ReLU	3	64	1	1
8	FCL1_Dropout				
9	FCL2_Softmax				

4.2 Statistical Analysis

Our 9-layer CNN with "BN-ReLU-AP-DO" was tested. Experiments ran ten times. As shown in Table 2, the average accuracy is 97.50 ± 1.65%. It can be seen that two runs achieve 100% accuracy and the overall accuracy rate is relatively high, which is benefit from less categories of classification, advanced techniques and data augmentation.

Table 2. Ten runs of our method

Run	Our method
1	95.83
2	100
3	95.83
4	97.92
5	97.92
6	95.83
7	100
8	97.92
9	95.83
10	97.92
Average	**97.50 ± 1.65**

5 Discussions

5.1 Effect of Batch Normalization and Dropout Layers

To validate the effect of batch normalization (BN) and dropout techniques, we compared CNN using BN and CNN without BN, furthermore, we compared CNN using dropout and CNN without dropout. As shown in Fig. 7, it was found that the accuracy decreased to 18.75% when we removed BN layers from neural network. This is a huge drop, which indicated that BN layer played a significant role in our method. Meanwhile, dropout technique can help to increase 1.67%, which denoted that dropout has a positive effect but is not obvious.

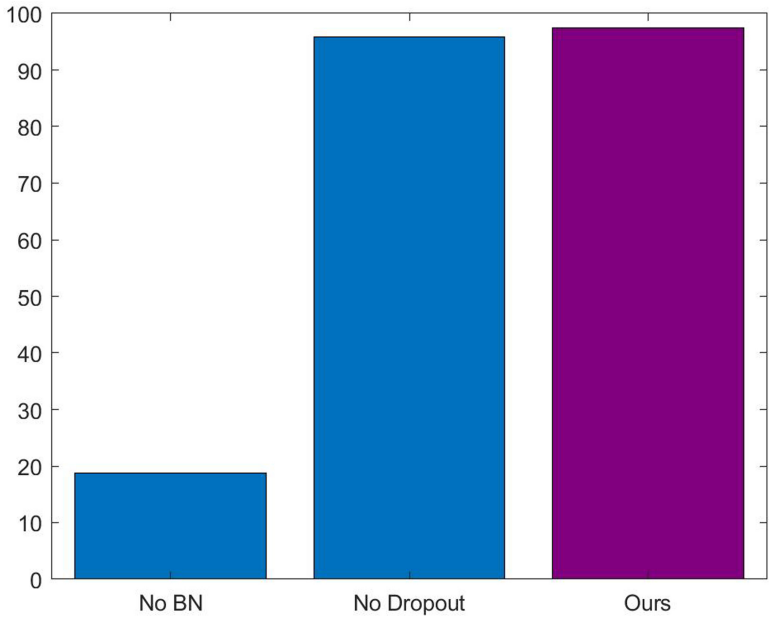

Fig. 7. Effect of batch normalization and dropout

5.2 Effect of Data Augmentation

We executed the same validation between with DA and without DA to confirm the effect of data augmentation. As listed in Table 3, not using DA will decrease the accuracy by 4.58%. Thus, data augmentation can provide sufficient image data and improve performance. We will test more DA techniques in the future than in this experiment.

5.3 Training Algorithms Comparison

In this experiment, three training algorithms i.e. Adam, RMSProp and SGDM were compared. The results were represented in Fig. 8. The means and standard deviation of

Table 3. Comparison of with DA and without DA

	With DA	Without DA
Average	97.50 ± 1.65	92.92 ± 1.07

Adam, RMSProp and SGDM are 97.50 ± 1.65%, 90.63 ± 1.77% and 91.25 ± 1.32%, respectively. It can be observed that Adam algorithm is significantly better than other two training algorithms.

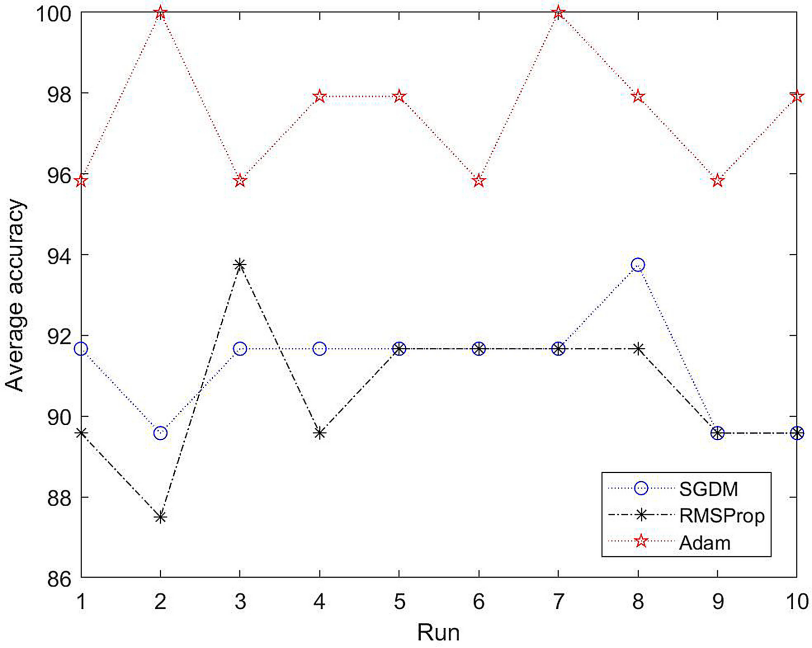

Fig. 8. Training algorithms comparison

5.4 Comparison to State-of-the-Art Approaches

In the experiment, we compared customized 9-layer CNN-BN-ReLU-AP-DO method with other three traditional machine learning approaches i.e. Decision Tree, KNN and SVM. The comparison can be observed in Fig. 9. In these four state-of-the-art approaches, average accuracy of Decision Tree, KNN, Quadratic SVM and CNN9 (ours) are 64.7%, 82.9%, 83.7% and 97.5%, respectively. Obviously, our method is superior to other state-of-the-art approaches.

The reason why we got the best performance in four approaches can be focus on three points. First, the introduction of BN can solve the issue of internal covariate shift

and enhance performance. Second, dropout technique and DA can help to overcome overfitting. Finally, Adam algorithm is benefit to accelerate learning speed and improve effectiveness.

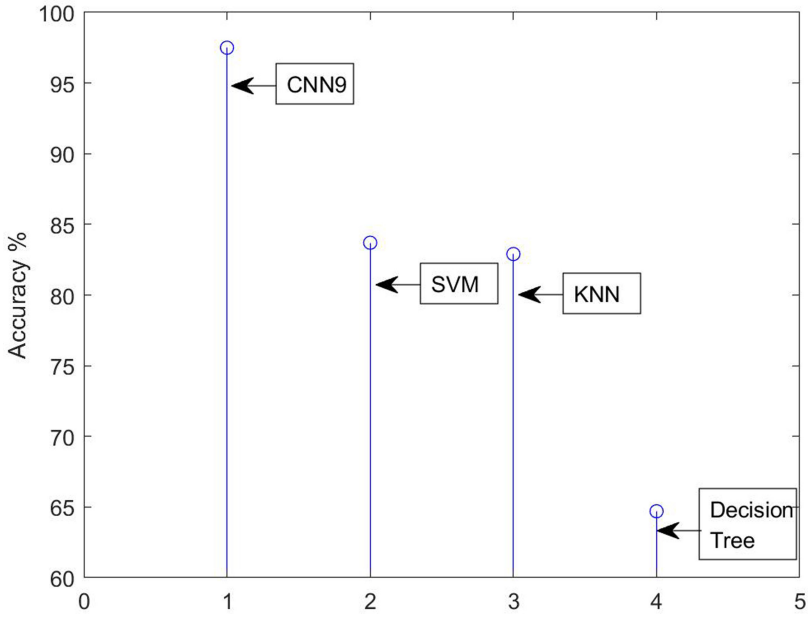

Fig. 9. Comparison to state-of-the-art approaches

6 Conclusions

A novel nine-layer "CNN-BN-ReLU-AP-DO" method was proposed for six-category similar gesture recognition. We trained this nine-layer CNN by Adam method, which achieves 97.50 $\pm$ 1.65% accuracy and is superior to other state-of-the-art approaches. Besides, we compared our method to CNN without BN and CNN without dropout. The results showed that BN technique played a major role and dropout provided a positive effect. Furthermore, data augmentation was proved to be beneficial for small dataset. Finally, different training algorithms were executed, It was found Adam is superior to RMSProp and SGDM.

In the future, we shall try other deep neural networks with different structures and optimize their hyperparameters [21]. Beside, we will transfer the mature method to other application field, such as facial expression recognition and lip language recognition.

Acknowledgements. This work was supported by Natural Science Foundation of Jiangsu Higher Education Institutions of China (19KJA310002), The Philosophy and Social Science Research Foundation Project of Universities of Jiangsu Province (2017SJB0668).

References

1. Lee, J.S., Lee, Y.J., Lee, E.H..: Hand region extraction and gesture recognition from video stream with complex background through entropy analysis. In: Proceedings of the 26th Annual International Conference of the IEEE EMBS, pp. 1–10 (2004)
2. Wu, X., Zhang, Q., Xu, Y.: A review of the development of gesture recognition research. Electron. Sci. Technol. **26**(6), 71–174 (2013). https://doi.org/10.3969/j.issn.1007-7820.2013.06.053
3. Jiang, L., Ruan, Q.: Research on gesture recognition technology based on neural network. J. Beijing Jiaotong Univ. **30**(6), 32–36 (2006). https://doi.org/10.3969/j.issn.1673-0291.2006.05.008
4. Feng, Z., Jiang, Y.: A review of gesture recognition research. J. Univ. Jinan (Sci. Technol.) **4**, 336–341 (2013)
5. Gao, Y., Jia, C., Chen, H., Jiang, X.: Chinese fingerspelling sign language recognition using a nine-layer convolutional neural network. EAI Endorsed Trans. e-Learn. **7**(20), e2 (2021)
6. Liu J, Zhao H. School of mechanical and electrical engineering and automation. J. Autom. EI CSCD, 31 (2020)
7. Zhang, Y., Li, L.: Realization of face feature point recognition based on cascaded convolutional neural network. J. Lanzhou Univ. Technol. **3**, 105–109 (2020)
8. Long, Y., Li, Y., Tao, W., et al.: Text sentiment analysis based on cascade convolution and attention mechanism. J. Taiyuan Normal Univ.: Nat. Sci. **2**, 30–36 (2020)
9. Jiang, X., Chang, L., Zhang, Y.D.: Classification of Alzheimer's disease via eight-layer convolutional neural network with batch normalization and dropout techniques. J. Med. Imaging Health Inform. **10**(5), 1040–1048 (2020)
10. Xiao, J., Tian, H., Zou, W.: Stereo matching based on convolution neural network. Inform. Technol. Inform. **38**(8), 0815017 (2018)
11. Yuan, M., Zhou, C., Huang, H., et al.: Survey on convolutional neural network pooling methods. Softw. Eng. Appl. **5**, 360–372 (2020)
12. Wang, S.Y., Teng, G.W.: Optimization design of ReLU activation function in convolutional neural network. Inform. Commun. **1673**(1131), 42–43 (2018)
13. M GCMMD: Noisy activation functions. arXiv preprint arXiv 1603:00391 (2016)
14. Jiang, A., Wang, W.: Research on optimization of ReLU activation function. Transducer Microsyst. Technol. **2**, 50–52 (2018)
15. Liu, J., Zhao, H., Luo, X., Xu, Y.: Research progress of deep learning batch normalization and its related algorithms. Acta Autom. Sinica **46**(6), 1090–1120 (2020)
16. Han, M.: Research and implementation of dropout method based on selective area drop, 003028, https://doi.org/10.27005/d.cnki.gdzku (2020)
17. Xie, F., Gong, J., Wang, Y.: Facial expression recognition method based on skin color enhancement and block PCA. J. Nanjing Normal Univ. (Eng. Technol. Ed.) **02**, 49–56 (2017)
18. Cai, L., Ye, Y., Gao, X., Li, Z., Zhang, C.: An improved visual SLAM based on affine transformation for ORB feature extraction. Optik **227**, 165421 (2021). https://doi.org/10.1016/j.ijleo.2020.165421
19. Wang, Y., Biyun, X., Kwak, M., Zeng, X.: A noise injection strategy for graph autoencoder training. Neural Comput. Appl. **33**(10), 4807–4814 (2020). https://doi.org/10.1007/s00521-020-05283-x
20. Zhang, X., Zuo, C., Shen, D.: Gamma nonlinear error correction method based on deep learning. G01B11/25, 1–10.
21. Jiang, X., Satapathy, S.C., Yang, L., Wang, S.-H., Zhang, Y.-D.: A survey on artificial intelligence in Chinese sign language recognition. Arab. J. Sci. Eng. **45**(12), 9859–9894 (2020). https://doi.org/10.1007/s13369-020-04758-2