An Optimized Eight-Layer Convolutional Neural Network Based on Blocks for Chinese Fingerspelling Sign Language Recognition

Huiwen Chu, Chenlei Jiang, Jingwen Xu, Qisheng Ye, and Xianwei Jiang(✉)

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

Abstract. Sign language plays a significant role in communication for the hearing-impaired and the speech-impaired. Sign language recognition smooths the barriers between the disabled and the healthy. However, the method has been difficult for artificial intelligence to use because it requires complex gestures that must be recognized in real time and with great accuracy. Fingerspelling sign language recognition methods based on convolutional neural networks have gradually gained popularity in recent years thanks to the advancement of deep learning techniques. Recognition of sign language using finger spelling has taken center stage. This study proposed an optimized eight-layer convolutional neural network based on blocks (CNN-BB) for fingerspelling recognition of Chinese sign language. Three different blocks: Conv-BN-ReLU-Pooling, Conv-BN-ReLU, Conv-BN-ReLU-BN were adopted and some advanced technologies such as bath normalization, dropout, pooling and data augmentation were employed. The results displayed that our CNN-BB achieved MSD of 93.32 ± 1.42%, which is superior to eight state-of-the-art approaches.

Keywords: fingerspelling · sing language recognition · batch normalization · data augmentation · pooling · dropout

1 Introduction

Sign language is one of the primary means of communication for the hearing-impaired and speech-impaired. It is a type of language that uses the shape of the hand as a carrier and simulates images or syllables by changing gestures to form certain meanings or words. As an independent visual language with a complete grammatical system, sign language in deaf people and disabled people occupy the core status in the field of communication. In April 2022, the China National Emergency Language Service Team was established in Beijing and put forward the proposal of "doing a solid job in providing emergency language services". As an important emergency language, sign language is naturally a key link in the construction of such work. Our research enthusiasm for sign language is growing as the country attaches great importance to it.

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2024
Published by Springer Nature Switzerland AG 2024. All Rights Reserved
B. Wang et al. (Eds.): ICMTEL 2023, LNICST 535, pp. 11–31, 2024.
https://doi.org/10.1007/978-3-031-50580-5_2

The major fields of sign language study include computer science and technology, linguistics, and special education. As the leading research field of sign language, computer science and technology mainly focuses on pattern recognition, signal processing, computer vision, visual language, etc., among which the most important research topic is "sign language recognition". A large number of scholars are immersed in studying various problems of sign language recognition technology and enriching various methods of sign language recognition [1].

Sign Language Recognition (SLR) is a technology that uses computers to convert gestures into text or speech information [2]. Traditional methods of sign language recognition include template matching, Hidden Markov (HMM), and NN, etc. The above conventional approaches have certain drawbacks of their own, but by incorporating innovation, they can overcome these drawbacks as science and technology continue to advance and grow. For example, combining HMM with Dynamic Time Warping (DTW) [3] or Support Vector Machine (SVM) [4] or NN [5], combining fuzzy logic with NN [6], etc.

In recent years, the rapid development of deep learning technology has brought new vitality to SLR. The current major deep learning-based sign language recognition technologies include sign language recognition technology based on Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Graph Neural Network (GNN) and the integration of various methods.

CNN is an important form of deep learning, which is dedicated to processing data with similar network structure, such as time series and image data. CNN has a huge impact on the field of Chinese sign language recognition because it is extremely effective at handling picture categorization difficulties [7]. Chinese sign language is divided into gesture sign language and fingerspelling sign language, the latter usually includes 30 fingerspelling forms of Chinese sign language: 26 letters (a–z), three retroflex 3 letters ("ch", "sh", and "zh"), and one nasal consonant ("ng"). Therefore, a sign language with only 30 letters is more precise [8] and simpler when facial emotions are excluded. Figure 1 shows the letters of the Chinese fingerspelling sign language [9].

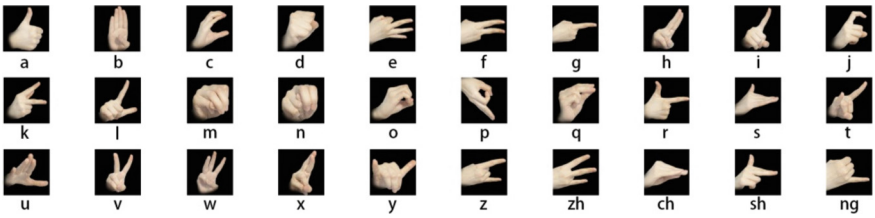

Fig. 1. Alphabets in Sign Language

An important scientific advancement is the improvement of Chinese sign language fingerspelling recognition. It can not only aid in the better social integration of the deaf but also advance the study and advancement of computer vision, artificial intelligence, and other related fields. An effective and precise method for recognizing sign language is finger-spelling recognition of Chinese sign language using convolutional neural networks. This method can be used to enhance the precision and effectiveness of artificial

intelligence and natural language processing in speech recognition, handwriting recognition, and other areas. Chinese Sign Language fingerspelling recognition has enormous application potential and will make people's lives and jobs more convenient.

In order to increase the precision of Chinese fingerspelling sign language identification, this article will install an eight-layer convolutional neural network that has been tuned and will also incorporate data augmentation. By integrating pooling, batch normalization, and dropout approaches, we also increase the validity of the test data, overcoming the shortcomings of pre-training and enhancing CNN usability and accuracy. This article's remaining sections are organized as follows: The data set is described in Sect. 2. Section 3 explicitly introduces the Chinese fingerspelling sign language recognition methods utilized in this study. The experiment technique is described in Sect. 4, comments are provided in Sect. 5, and conclusions are provided in the last section.

2 Dataset

We collected several groups of Chinese finger gesture image samples to establish a relevant gesture data set, each sample covered 26 letters plus "zh", "ch", "sh", "ng" two-syllable finger language pictures, a total of 30, using the brush tool of Photoshop, set the soft light mode, smear the original image brightness, to ensure that the image is clearly visible processing. In addition, multiple samples take into account individual differences in the use of sign language gestures, which makes the research results more convincing (See Fig. 2).

Fig. 2. The various sign language gestures for the letter "zh"

3 Methodology

The image modification in this paper is based on deep neural network. For function extraction of input data, image feature processing and noise reduction are some of the main uses of convolutional layer. By using ReLU function (improve data processing speed), Pooling layer (reduce the number of neural network parameters and computing load to improve computing speed), Dropout (prevent over-fitting, It provides an effective combination of exponential neural network architecture method), Batch normalization (accelerating the speed of model training and increasing the ability of generalization), and Data augmentation (increasing the size of data in some ways to improve the performance of its learning algorithm) to improve the performance of convolutional networks. It is convenient for our experimental calculation, as shown in Fig. 3.

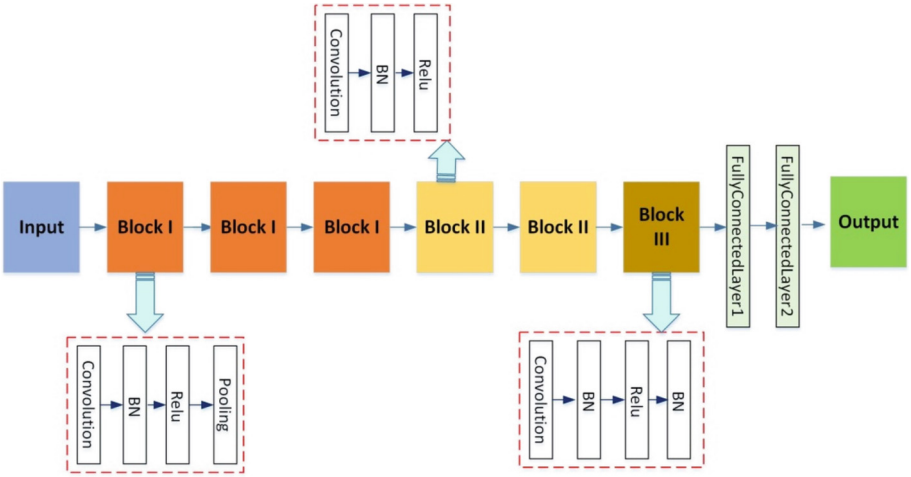

Fig. 3. The proposed network architecture

3.1 Convolutional Layer

Neural network, also known as artificial neural network or analog neural network, is a network built by connecting a large number of artificial neurons in different ways. It is a feedforward model that mimics the structure and function of signals transmitted by biological neurons, so that it can have simple decision ability and simple judgment ability like people. CNN, Convolutional Neural Network, is a feedforward neural network in the deepening research of neural networks. Its artificial neurons can respond to a subset of surrounding units within a coverage area, making it even better at large-scale image processing. LeCun (1989) first used the term "convolution" [10] when discussing its network structure, hence the name "convolutional neural network".

The basic structure of CNN is the input Layer, the Convolutional Layer, the Max Pooling Layer (also known as the sampling layer), the Fully Connected Layer and the output layer. As shown in the Fig. 4, the image is firstly searched for features by the convolution layer, then downsampling is conducted by the downsampling layer, some information is ignored to reduce the training parameters while the sampling remains unchanged, and finally classification judgment is made by the full connection layer. In CNN, the neuron input value is the local connection between the neuron and the input on the input feature plane of the convolutional layer, and the corresponding connection weight is obtained by adding the weighted offset value of the local input. This process is equivalent to the convolutional process [11]. Generally, the convolutional layer and the subsampling layer are not single, and they are usually set alternating. It's a convolution layer connected to the lower sampling layer connected to the convolution layer connected to the lower sampling layer and so on. Sample Heading (Third Level). Only two levels of headings should be numbered. Lower level headings remain unnumbered; they are formatted as run-in headings.

In the convolutional layer, the feature graph of the upper layer is convolved by a learnable filter (i.e. convolution kernel), which circulatively convolves the entire input

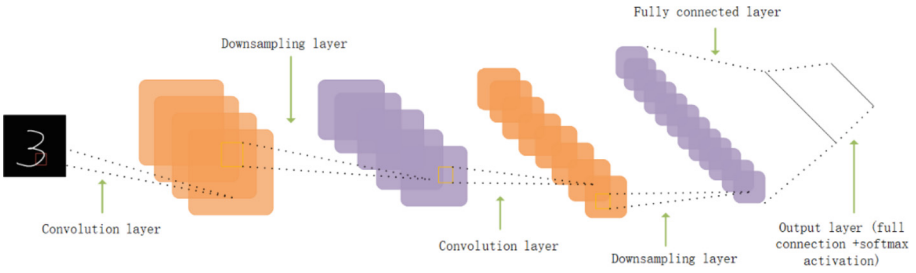

Fig. 4. CNN operation process

image with a certain step size in the original image, and then obtains the output feature graph through an activation function. Each output feature graph can be convolved with the values of multiple feature graphs:

$$x_n^p = f\left(y_n^p\right) \tag{1}$$

$$y_n^p = \sum x_m^{p-1} * k_{mn}^p + b_n^p (m \in M_n) \tag{2}$$

Among them, the net activation of the n TH channel of the convolution layer l (y_n^p), the output of the jTH channel of the convolution layer I (x_m^{p-1}), the activation function ($f(\cdot)$), the input feature map subset of calculating y_n^p (M_n), the convolution kernel matrix (k_{mn}^p), the bias of the convolution eigengraph (b_n^p), the convolution symbol($*$).

The formula of output feature graph for each feature graph in the lower sampling layer is as follows

$$x_n^p = f\left(y_n^p\right) \tag{3}$$

$$y_n^p = \gamma_n^p down\left(x_m^{p-1}\right) + b_n^p \tag{4}$$

The net activation of the nTH channel in the lower sampling layer l(y_n^p), the weight coefficient of the lower sampling layer (γ), the offset term of the lower sampling layer (b_n^p), the downsampling function ($down(\cdot)$).

Each layer in the fully connected layer is a tiled structure composed of many neurons, essentially a perceptron that classifies or regress the input data, and whose output can be obtained by a weighted summation of the input and by a corresponding activation function.

$$x^p = f\left(y^p\right) \tag{5}$$

$$y^p = \varphi^p x^{p-1} + b^p \tag{6}$$

The net activation of the fully connected layer l(y^p), the weight coefficient of the fully connected network (φ^p), the offset item of the fully connected layer (b^p) [10, 12].

CNN, due to incomplete connections between neurons, the sharing of connection weights of silent neurons at the same layer is a special difference from other deep neural network models. These special points make it closer to the biological neural network, and the complexity, parameter training and weight number of the network model are further reduced. In the previous deep neural network, the overfitting problem of the model has been greatly alleviated, and the memory occupied by the model has been clearly reduced At the same time, CNN also has strong stability and fault tolerance, which can show the advantages of efficiency and accuracy in the task of image recognition and influence classification [13].

3.2 ReLU Function

In deep neural network learning, activation function plays an important role in stimulating hidden nodes to produce better output. The main purpose of activation function is to introduce nonlinear features into the model. The commonly used activation functions include Logistic sigmoid and tanh. Hahnloser et al. introduced ReLU function into a dynamic network for the first time in 2000, and proved for the first time in 2011 that ReLU function can train a deeper network better than previous ones. Up to now, ReLU is still the most commonly used activation function for the execution of most deep learning tasks, and it is also widely used by the excitation layer of CNN.

Rectified Linear Unit Relu (Rectified Linear Unit) is a commonly used activation function in artificial neural networks. Generally speaking, it refers to a slope function in mathematics. The definition is as follows (Fig. 5):

$$\boldsymbol{ReLU} = \boldsymbol{max}(\mathbf{0}, \boldsymbol{y}) \tag{7}$$

$$\boldsymbol{y} = \begin{cases} \boldsymbol{x}, \ \boldsymbol{if}\ \boldsymbol{x} > \mathbf{0} \\ \mathbf{0}, \ \boldsymbol{if}\ \boldsymbol{x} \leq \mathbf{0} \end{cases} \tag{8}$$

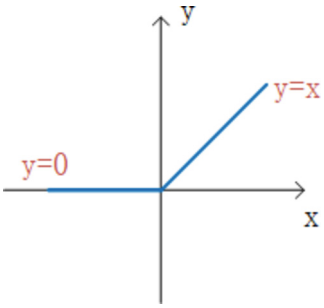

Fig. 5. The ReLU function

Compared with other activation functions, ReLU function and its derivatives are relatively simple in mathematical operation, short in calculation time, fast in speed, occupy resources, and have the advantages of imitating biological principles, so they are basically

the preferred type when facing unknown data sets. And for linear function, its expression ability is stronger, especially in the deep network; For nonlinear functions, Since the gradient of ReLU's non-negative interval is constant, the gradient always exists, so the convergence rate of the model is stable. Of course, ReLU function also has its defects. The slope of its negative end is 0, and the convolution output data cannot be expressed in the negative part, which also leads to the failure of neurons to update parameters, the slow processing of data sets, and the waste of computing resources occupied by neurons that do not update parameters [14]. ReLU is an asymmetric piecewise function. If it is used to fit a smooth nonlinear function, it requires more neurons or hidden layers with the same fitting accuracy. And the indifferentiability of the ReLU function at zero [15]. However, since zero differentiability is still a special case, and various variants have been developed by researchers to make up for its deficiencies, such as Softplus, Leaky ReLU, ELU, SiLU, etc. to improve the performance of some tasks, ReLU is by far the best activation function of deep learning.

3.3 Pooling Layer

In view of the large size of image and convolutional feature graph, we do not need to deal with too much redundant information in practical application. The key is to extract image features. Therefore, we adopted a strategy similar to image compression for optimization. Then how to carry out this kind of compression idea? The pooling layer plays such a role. In general, the pooling layer of convolutional neural network follows the convolutional layer, and the operations in the pooling layer are called pooling operations.

Pooling principle is similar to the principle of convolutional layer. Convolution is a kind of linear operation algorithm about matrix, and its working mode is to carry out sliding windowing operation on input matrix and convolution kernel to realize feature extraction of image. Meanwhile, the pooling layer also adopts the idea of overlapping Windows to divide input data into multiple blocks for pooling operation to realize feature extraction of image. Common pooling operations include maximum pooling and average pooling. Maximum pooling takes the maximum value of the local area of the original image as the output of this area, and average pooling takes the mean value of the local area of the original image as the output of this area.

Pooling steps are relatively fixed. Pooling steps are relatively fixed. In the first step, we set the pooled window size and step size, which are generally set to the same value. For example, set the step of the 8×8 pool window to 8, and the step of the 4×4 window to 4. The second step is to move the pooling window on the input matrix. For each region, there is an operation result to represent the feature element of the region. Step 3 Repeat Step 2.

For a 4×4 image to be processed, define a 2×2 pooling window step size set as 2, do the maximum pooling operation, move 2 steps each time, the moved subareas cannot be overlapped, and the image is finally compressed into 1/4 of the original. As you can imagine, the 2×2 window is constantly moving from left to right or top to bottom from the input image, constantly mapping the eigenvalues in this area from the window. After traversing the original image, it will get an image that represents the pool result and is smaller than the original image. The image's dimensions are decreased by 75% (Fig. 6).

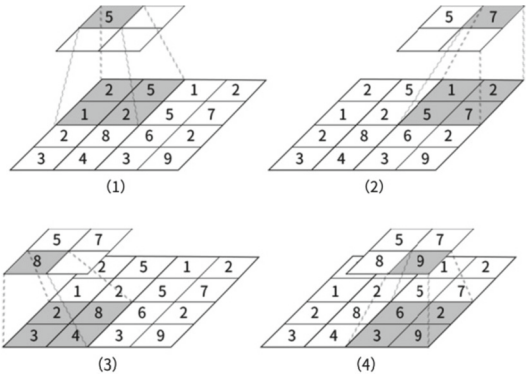

Fig. 6. Maximum pooling with window size 2 and step size 2

Average pooling is similar, except that the output of the 2×2 pooled subwindow size uses average pooling to represent the characteristic elements of that subarea.

According to the implementation method of pooling operation, it can compress the image size of the input feature map. On the premise that the image can be recognized, the depth dimension remains unchanged and unnecessary redundant information of the image is reduced, which reflects the feature invariance in image processing. For example, when the resolution of a photo of a tree is lowered, we can still recognize it as a tree, indicating that the photo still retains the characteristics of a tree. At the same time, the compressed image pixel matrix is greatly reduced, reducing the number of neural network parameters and computing load, and improving the computing speed.

3.4 Batch Normalization

In most cases, there are many more layers of deep neural network than we expect. When data passes each layer of neural network and activation function, Internal Covariate Shift occurs [16]. By adding a Batch Normalization (BN) algorithm to the appropriate network layer, it keeps it from fluctuating too much, keeping sample output values for that layer within given ranges (Fig. 7).

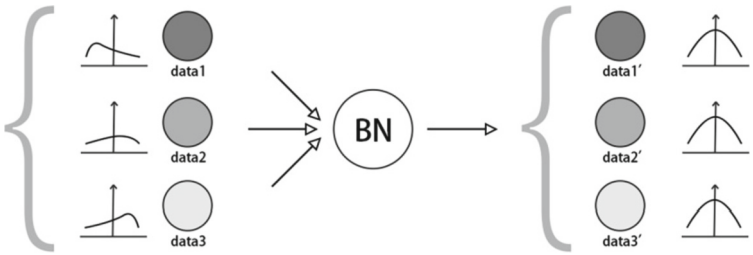

Fig. 7. Batch normalization principle

Batch Normalization, also known as batch normalization, is a data preprocessing tool used to adjust numerical data to common proportions without distorting its shape. In general, when we are in a deep learning algorithm, we tend to change the value to a balanced ratio. Normalization is to speed up model training.

In general, BN layer is placed in the convolution, it is to solve the neural network training "gradient dispersion", and "gradient explosion" problem of important technical means. Batch Normalization enables data normalization of batches of sample data in a specific way. Normalization selects a part of data from the network layer as sample input, subtracts its mean value from the input value and divides it by the standard deviation of the data, and then completes data preprocessing. But why only select a subset of data from the network layer and normalize it? Assuming that all the data of a layer are normalized, it will produce huge computing overhead and lose the significance of optimization.

If the network layer m receives batch inputs, each node of the layer generates m outputs during forward propagation. Batch Normalization aims to normalize these m outputs at each node of the layer. This normalization is calculated as follows.

$$\mu_B \to \frac{1}{m}\sum_{i=1}^{m} x_i \tag{9}$$

$$\sigma_B \to \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2 \tag{10}$$

$$\widehat{x_i} \to \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{11}$$

However, the normalization of input nodes in each layer may change the representation of the data in that layer. Because normalization processing turns one set of data into another set of data, each set of data contains different information, so it cannot be finished after normalization processing. It is also necessary to process the linear transformation of the normalized data to get the final result of the linear transformation ζ_i. . The formula is as follows (12).

$$\zeta_i = \zeta_i \cdot \widehat{x_i} + \lambda_i \tag{12}$$

It is worth mentioning that ζ and λ can still play a role in restoring data expression ability to some extent. Suppose a feature x in a certain network layer, the mean value obtained by mini-batch calculation of this feature is u, and the standard deviation is o. The normalization of x gives you t, and the linear transformation gives you y. So there's a special result, when y is equal to o and beta is equal to u the result of the linear transformation y is exactly equal to the characteristic x before the normalization. You can guess the purpose of the formula, and you can reconstruct the meaning of the network layer data, but there are very few such coincidences, because y and beta are trained by the model.

BN brings three benefits. First, it speeds up model training and improves learning rate. Second, it has certain regularization effect. The use of BN increases generalization capabilities, even without Dropout, and reduces the use of L2 regularization; Third, there

is an opportunity to make the model work better. This effect is not absolute, but many models do get better with BN.

Although BN has many advantages, it can also cause problems if it is not used properly. For example, Batch statistical estimation is not accurate will lead to batch smaller, batch normalization error will increase rapidly. Therefore, I also need to know its scope of application. First, batch normalization can be added to the general neural network training to speed up the training. Second, it is suitable for the scenario where each mini-batch is large and the data distribution is similar.

3.5 Dropout

The term "Dropout" [17] was introduced by author Hinton in the paper "Improving neural networks by preventing co-adaptation of feature detectors". The phenomenon of "data overfitting" usually results from training complex feedforward neural networks in small data sets. One of the existing solutions is to improve the performance of neural networks by preventing the co-action of feature detectors, which is used by Alex and Hinton in their paper "ImageNet Classification with Deep Convolutional Neural Networks". Dropout algorithm has good effect on avoiding overfitting. Moreover, the "AlexNet network model" mentioned in this paper leads the trend and makes CNN the core algorithmic model for image classification [18].

During Dropout training of deep neural networks, overfitting is reduced by ignoring a portion of the feature detectors (counting the corresponding hidden layer node values as 0). This approach reduces interactions between feature detectors (hidden layer nodes), where detector interactions are those where some detectors depend on other detectors to function.

Dropout is a regularizer against overfitting. It is a regularization method that randomly sets the activation of the hidden units of each training case to zero at training time. This breaks the co-adaptation of the feature detector, because the exiting units cannot affect the other retained units. In other words, Dropout creates an efficient form of model averaging in which the number of trained models is exponentially related to the number of units and these models share the same parameters. Dropout also inspired other stochastic model averaging methods such as random pooling. Dropout works well in the fully connected layer of convolutional neural networks [19].

There is a trained neural network as follows, as shown in Fig. 8.

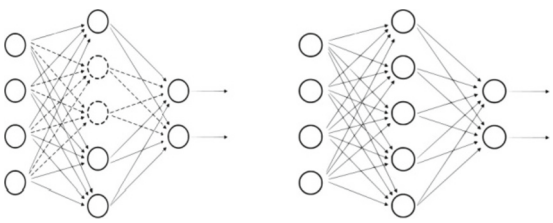

Fig. 8. Standard neural networks and partially temporarily deleted neurons

The flow of the diagram on the left is: First propagate the input quantities forward through the network, and then back propagate the errors to decide how to update the parameters for the network to learn.

(1) Randomly delete a portion of the neurons in the hidden layer of the network. This operation is temporary and the input and output neurons remain unchanged (the dashed graph in Fig. 8 indicates the temporarily deleted neurons).
(2) Propagate the input quantities forward through the modified network, and then propagate the obtained loss results backward through the modified network. After performing this process on the training samples, the corresponding parameters need to be updated on the neurons that have not been deleted according to the stochastic gradient descent method.
(3) Restore the deleted neurons (the deleted neurons are not updated at this time, while the remaining neurons have been updated) by selecting a random subset of the hidden layer neurons of the same size as in (1) and temporarily deleting them (the parameters of the deleted neurons need to be backed up). For the training samples, forward propagation and then backward propagation of the loss is performed and the parameters are updated according to the stochastic gradient descent method (the parameters of the deleted neurons remain unchanged and the remaining parameters are updated). The above process is repeated continuously.

The selection scheme of which units to discard is random. Each unit is guaranteed to be retained with a fixed probability independent of the other units, where the fixed probability is chosen in two ways, either based on the validation set or set directly to 0.5 or 0.3.

Dropout is being heavily used in fully connected networks and less used in the hidden layer of convolutional networks. The specific reasons include features such as sparsification of convolution itself. In general, Dropout is a hyper-parameter, not the parameter of each layer inside the general model network structure, but the parameter that needs to be adjusted artificially to try to improve the model effect according to the actual network, the actual application area.

3.6 Data Augmentation

'Data augmentation' is a set of techniques that artificially augment a data set by modifying a copy of existing data or using a collection of existing data into a new copy of the generated data set. It acts as a regularization and reduces overfitting when training machine learning models. If we have 50 experimental image data sets, we can create new copies of the images and double the training set by flipping them randomly horizontally and vertically.

To avoid the problem of overfitting, data enhancement techniques need to be used wisely to improve the performance of the algorithm. Yu Gao et al. proposed a data preprocessing method based on the convolutional neural network model Alex Net, using dataset augmentation, background segmentation and principal component analysis on the dataset. The original public dataset Leaves and the apple surface lesion dataset were firstly tested for classification and recognition. The results show that the recognition accuracy of both the public dataset Leaves and the apple surface lesion dataset on this network

has been improved after data augmentation [20]. For the traditional data-driven transient stability analysis method of power system, Yan Zhou et al. proposed a transient stability prediction method based on data augmentation and deep residual network considering the impact of the input data with noise and missing information on the performance of the prediction model. A special convolutional neural network-depth residual network in image processing is used to construct a deep model for transient stability evaluation by using dynamic data of the generator after perturbation as input features [21]. Our dataset is to include different conditions such as different orientations, positions, scales, brightnesses, etc. However, during the actual data collection, the number of datasets we collect is limited. By performing data augmentation and collecting a large number of sample data, we can solve the problem of sample data and prevent the neural network from learning irrelevant features and fundamentally improve the overall performance [22].

In the case of image enhancement, you can randomly flip, crop, rotate, scale, resize, stretch, and imitate. In addition, you can change saturation brightness, contrast, sharpness, and even add noise.

In the case of image enhancement, you can randomly flip, crop, rotate, scale, resize, stretch, and imitate. In addition, you can change saturation brightness, contrast, sharpness, and even add noise. Stable convolutional neural networks can correctly classify objects in different situations, and, increasing the trained data can improve the performance of the CNN model. The CNN is invariant to transition, viewpoint, size and illumination, and it works with 'Data augmentation'. In the article, we exemplify the following six examples of 'Data augmentation'.

3.6.1 PCA Color Enhancement Method

The algorithm performs principal component analysis based on the color channels and adds the color distribution of the original image to perform 'Data Augmentation'. The PCA color augmentation method is mainly used to change the brightness, contrast and saturation of the image. In order to maintain valid data such as relative color differences, major color families, and contours of the artificial image, principal component analysis is required for the training data set to recover the colors of its distribution principal axes [23]. Then, the artificial image is created by continuously adjusting the multiplicity of principal components of the dataset.

3.6.2 Noise Injection

There can be many kinds of noise injection in neural networks, such as input layer, hidden layer, weights, output layer, etc. The core of noise injection is to randomly disturb each pixel RGB of an image by adding a matrix of random values sampled from a Gaussian distribution to produce some new noise-contaminated image. It also augments the dataset and improves the ability to fit the true distribution of the data, helping the CNN to learn more powerful [24].

3.6.3 Scaling

The scaling methods include inward or outward scaling. When scaling inward, the size of the newly generated image becomes smaller; when scaling outward, the size of the newly generated image becomes larger. The image frame is a piece of the newly generated image that is equal in size to the original image.

3.6.4 Random Transfer

Random shifting only involves moving the image in the X or Y direction (or both), and when shifting we need to make assumptions about boundaries. With this enhancement method, most objects can be located almost anywhere in the image, so the convolutional neural network can recognize all the corners as well. Using this method can be very effective in enhancing the amount of data if the image has a monochrome background or a pure black background. Random flipping includes horizontal and vertical flipping. Of these, horizontal flipping is the most commonly used, but depending on the actual target, vertical flipping and other angles of flipping can also be used.

3.6.5 Gamma Correction

The nonlinear photoelectric conversion characteristics of sensors in electronic devices (e.g., camcorder, monitor) require the application of Gamma correction, which edits the Gamma curve of an electronic image and then performs nonlinear tonal editing of the image [25]. The basic idea of Gamma correction is to segment each of the color spaces R, G, and B and use linear functions in each segment to correct. This series of linear functions is generated based on the compensation of the Gamma curve by using a series of linear functions instead of a symmetric curve of the Gamma curve about the function $y = x$. Gamma represents a diagonal line between the output and input values of the image, with a gamma value of usually 2.3.

Gamma correction has a significant effect on the image, and different Gamma curves can achieve different results. The contrast of the whole image is related to the Gamma correction, and the higher the contrast, the more obvious the visual effect to the human eye. The color of the image is also related to the Gamma correction, the higher the contrast, the higher the color saturation of the whole image.

3.6.6 Affine Transformation

The affine transformation means that the image can be translated and rotated by a series of geometric transformations, while maintaining the flatness and parallelism of the image. Straightness means that a straight line is still a straight line and a half-circle is still a half-circle after the affine transformation; parallelism means that parallel lines are still parallel lines after the affine transformation.

Affine transform (AFT) is a permutation operation that randomly changes the position of image pixels. The affine transform of an image $f(x, y)$ The size of the $n \times n$ The affine transform of an image of pixels is calculated by $\left(x', y'\right)$, and the affine transform can be computed by left multiplying the original vector $\begin{bmatrix} x \\ y \end{bmatrix}$ by left multiplying one or

more transformation matrices, and multiple transformation matrices can be combined. The function is represented as follows.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = AFT\{(x, y), n\} = \left\{ \begin{bmatrix} a \\ b \end{bmatrix} + \begin{bmatrix} i & 0 \\ 0 & j \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \right\} (mod\ n) \tag{13}$$

where "mod" stands for modal operation, the $a \in [1, n]$, $b \in [1, n]$, the a, b the values of can be taken randomly in the range. i, j is determined by their relative to n the relative primes chosen. This i and j are chosen such that the affine transformation maps the original coordinates (x, y) mapped to a unique pixel in the transformed coordinates. If i and j are not relatively prime with respect to n are relatively prime, the affine transform maps different original coordinates to the same pixel in the transformed coordinates. After the AFT, the total energy of the input image remains constant [26].

4 Experiment Results

4.1 Experiment Configuration

On a personal computer running Windows 10 with a 2.5 GHz Intel Core i7 CPU and 32 GB of RAM, the experiment was carried out. The tests were carried out more times to smooth the randomness issue, and the overall accuracy was introduced to evaluate the results. We set the main parameters of training configuration as follows: Maximum Epochs was defined as 30, Initial Learn Rate was set to 0.01, Mini Batch Size is 256 and Learn Rate Drop Factor is 0.1.

4.2 Structure of Eight-Layer CNN Based on Blocks

In this paper, we introduced an eight-layer convolutional neural network with hybrid modules for Chinese fingerspelling sign language recognition. AS shown in the Fig. 3, the proposed network contains six convolutional layers with hybrid modules and two fullyconnected layers. Among them, the hybrid module is further divided into three situations: Block I (Conv-BN-ReLU-Pooling), Block II (Conv-BN-ReLU) and Block III (Conv-BN-ReLU-BN). The first hybrid mode is a commonly used combination, which is employed to verify and test the configuration of pooling. The second is a comparison mode, which omits the pooling operation. The third is a bold innovation. Here, the BN operation is applied twice to better standardize the input of adjacent layer and make the distribution more balanced and reasonable. All advanced technologies play their roles in their respective modules, and different combinations and cofigurations make the overall performance improved. The hyperparameters of proposed network have been demonstrated in Table 1. At the same time, the value of "Padding" is set to "same" and the dropout rate is 0.4.

4.3 Statistical Results

Our method CNNBB adopted 3 hybrid blocks was executed 10 runs, and the results are demonstrated in Table 2. As can be seen, the bolded portion of the column indicates

Table 1. Parameters of each layer based on blocks

Index	Layer	Filter Size	Filters	Stride
	Input			
1	Layer1-Block I	3×3	16	2
2	Layer2-Block I	3×3	32	2
3	Layer3-Block I	3×3	64	2
4	Layer4-Block II	3×3	128	2
5	Layer5-Block II	3×3	128	2
6	Layer6-Block III	3×3	256	2
7	FullyConnectedLayer1			
8	FullyConnectedLayer2			
	Output			

that the MSD (mean and standard deviation) is $93.32 \pm 1.42\%$. The maximum accuracy, however, comes in at 96.48%, while the lowest is 91.41%. All the accuracy values exceed 90%. Thus, it indicates that our method owns better stability and effectiveness.

Table 2. Ten runs of our method

Run	Accuracy of Our Method
1	92.97%
2	92.19%
3	91.41%
4	92.58%
5	92.19%
6	93.75%
7	93.75%
8	93.75%
9	96.48%
10	94.14%
MSD	**$93.32 \pm 1.42\%$**

5 Discussions

5.1 Comparison of Pooling Method

In this experiment, both maximum pooling (MP) and average pooling (AP) were verified without changing the parameter settings. As shown in Table 3, the results of 10 runs of average pooling are as follow: 90.06%, 91.23%, 89.45%, 91.28%, 91.84%, 92.02%, 91.45%, 92.23%, 91.63% and 93.58%. The MSD of average pooling is 91.48 ± 1.14%, which is litter lower to maximum pooling 93.32 ± 1.42%. Figure 9 is represented a vivid comparison between AP and MP. It is obvious from the results that maximum pooling performs well in recognition accuracy. It achieved the highest accuracy rate of 96.48%, while the average pooling rate was only 93.58%. In addition, the accuracy of maximum pooling is significantly better than that of average pooling each time of execution.

Table 3. Comparison of pooling method

Run	Average Pooling	Maximum Pooling
1	90.06%	92.97%
2	91.23%	92.19%
3	89.45%	91.41%
4	91.28%	92.58%
5	91.84%	92.19%
6	92.02%	93.75%
7	91.45%	93.75%
8	92.23%	93.75%
9	91.63%	96.48%
10	93.58%	94.14%
MSD	**91.48 ± 1.14%**	**93.32 ± 1.42%**

5.2 Effect of Double Bath Normalization

As an innovation, a double BN structure (Block III) was introduced to verify whether the overall performance can be improved. In this experiment, the Block III was placed to last position and it played a significant role. As can be seen, the comparison between using double BN and without double BN is indicated in Fig. 10 and Table 4. It denoted that double BN make the input layer more distributed and normalized, which can further avoid gradient disappearance and accelerate learning convergence.

5.3 Comparison to State-of-the-Art Methods

In this experiment, eight state-of-the-art methods: HMM [27], CSI [28], HCRF [29],GLCM-PGSVM [30], WE-KSVM [31], 6L CNN-LRELU [32], AlexNet-DA-Adam [33], CNN7-DA [34] were compared with our proposed network CNN-BB. As

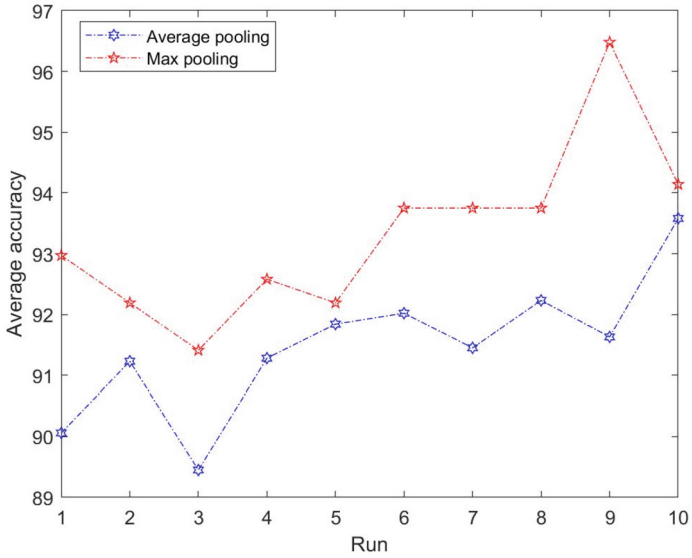

Fig. 9. Comparison of average pooling and maximum pooling

Table 4. Comparison of BN

Run	Singer BN	Double BN
1	91.41%	92.97%
2	89.06%	92.19%
3	91.80%	91.41%
4	90.63%	92.58%
5	91.41%	92.19%
6	91.41%	93.75%
7	92.19%	93.75%
8	92.58%	93.75%
9	92.58%	96.48%
10	91.41%	94.14%
MSD	**91.45 ± 1.03%**	**93.32 ± 1.42%**

can be seen in Fig. 11, our CNN-BB methods achieved superior MSD. Two important factors boost the discriminative ability of our network. Singer BN can provide the balance distribution of input and smooth the disappearance of gradient. Double BN can improve the effectiveness and accelerate the convergence of learning. Additionally, operation of pooling can decrease the computation and cut down the ovefitting. Meanwhile, dropout and ReLU also play their role in the architectures.

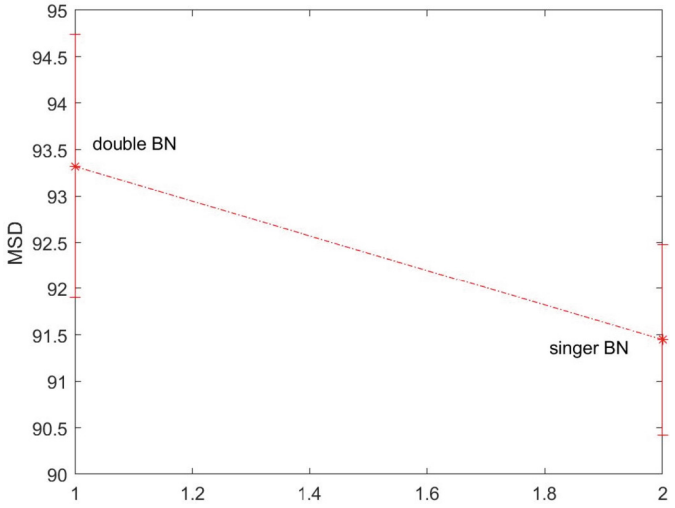

Fig. 10. Effect of Double Bath Normalization

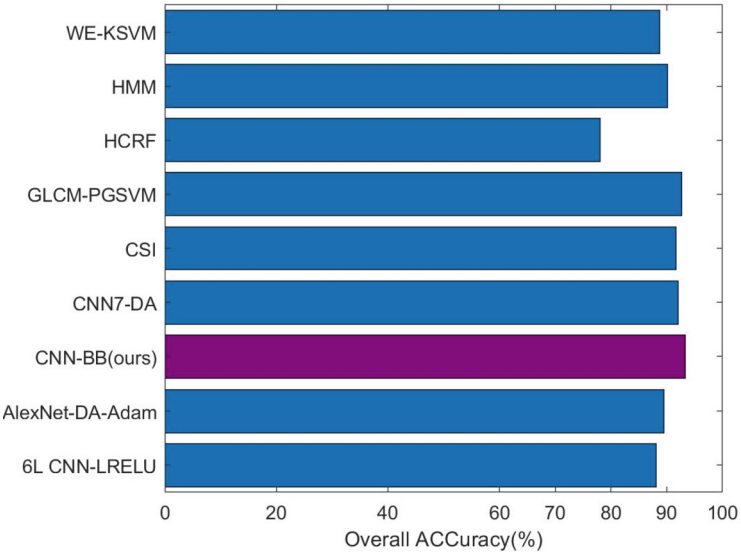

Fig. 11. Comparison to state-of-the-art approaches

6 Conclusions

In the above study, we proposed a block-based eight-layer optimized convolutional neural network (CNN-BB) for Chinese handwritten sign language recognition. In the architectures, three different blocks, that is, Conv-BN-ReLU-Pooling, Conv-BN-ReLU, Conv-BN-ReLU-BN were employed. Adopting drop-out, data enhancement, ReLU, batch

normalization and pooling and other advanced technologies, the CNN-BB proposed in this paper achieved 93.32 ± 1.42% MSD, which is superior to the other advanced method.

In future studies, we will try to establish more adequate data sets, more advanced methods and fine-tune the hyperparameters in the hope of obtaining better performance. Meanwhile, transfer learning and shifting the optimized network to other fields are also the goals we should focus on.

Acknowledgements. This work was supported by National Philosophy and Social Sciences Foundation (20BTQ065), Natural Science Foundation of Jiangsu Higher Education Institutions of China (19KJA310002).

References

1. Yang, X., Lei, J., Sun, K.: Evolution and trend of sign language research in China: a visual analysis based on CiteSpace, vol. 267, no. 09, pp. 21–28+65 (2022)
2. Yu, Z.: Adaptive problems in Chinese sign language recognition, Ph.D. Harbin Institute of Technology (2010)
3. Yao, G., Yao, H., Jiang, F.: A multi-layer classifier sign language recognition method based on DTW/ISODATA algorithm, vol. 08, pp. 45–47+200 (2005)
4. Zhao, W.: Chinese sign language recognition based on HMM_SVM, vol. 21, no. 10, pp. 24–26 (2011)
5. Wu, J., Gao, W.: ANN/HMM based sign language recognition method, no. 10, pp. 63–66 (1999)
6. Zou, W., Yuan, K., Du, Q., Xu, C.: Fuzzy neural network based word recognition in static sign language, no. 04, pp. 616–621 (2003)
7. Ma, C., Shao, J., Qin, B.: Progress in sign language recognition in the teaching of the hearing impaired, vol. 42, no. 10, pp. 23–27 (2022)
8. 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)
9. Lee, Y., Hua, F.: Principle and realization of conversation from standard Chinese pinyin to international phonetic alphabet, vol. 14, pp. 540–545 (2012)
10. Feng, B., Yang, H., Yuan, G., Li, J., Zhan, C.: A review of the research of neural networks in SAR image target recognition, vol. 42, no. 10, pp. 15–22 (2021)
11. Zhou, F., Jin, L., Dong, J.: Review of convolutional neural networks, vol. 40, no. 06, pp. 1229–1251 (2017). https://kns.cnki.net/kcms/detail/11.1826.TP.20170122.1035.002.html
12. Chang, L., et al.: Convolutional neural networks in image understanding. Acta Autom. Sin. **42**(09), 1300–1312 (2016). https://doi.org/10.16383/j.aas.2016.c150800
13. Zhang, Y., Liu, Y., Liu, M., Man, W., Song, T., Li, C.: Fine classification of wetland plant communities based on relief F and convolutional neural networks, no. 02, pp. 58–64 (2023). https://doi.org/10.13474/j.cnki.11-2246.2023.0041
14. Qian, X., Zhang, X., Hao, Z.: Gait recognition based on improved convolutional neural network, vol. 9, no. 02, pp. 91–97 (2022). https://doi.org/10.19306/j.cnki.2095-8110.2022.02.011
15. Hao, T.: Construction of activation function LeafSpring and comparative study of multiple data sets, vol. 49, no. 03, pp. 306–314+322 (2020). https://doi.org/10.13976/j.cnki.xk.2020.9332

16. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. Presented at the Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, Lille, France (2015)
17. Hinton, G.E., Srivastava, N., Krizhevsky, A., et al.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv 2012; abs/1207.0580
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Commun. ACM **60**, 84–90 (2012)
19. Srivastava, N., Hinton, G., Krizhevsky, A., et al.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
20. Gao, Y., Zhou, B., Hu, X.: Research on convolutional neural network image recognition based on data enhancement. Comput. Technol. Dev. **28**, 62–65 (2018)
21. Yanzhen, Z., Xiangyu, C., Jian, L., et al.: Transient stability prediction of power systems based on data augmentation and deep residual networks. China Electr. Power **53**, 22–31 (2020)
22. Eckert, D., Vesal, S., Ritschl, L., Kappler, S., Maier, A.: Deep learning-based denoising of mammographic images using physics-driven data augmentation. In: Tolxdorff, T., Deserno, T., Handels, H., Maier, A., Maier-Hein, K., Palm, C. (eds.) Bildverarbeitung für die Medizin 2020. Informatik aktuell, pp. 94–100. Springer, Wiesbaden (2020). https://doi.org/10.1007/978-3-658-29267-6_21
23. Vasconcelos, C.N., Vasconcelos, B.N.: Convolutional neural network committees for melanoma classification with classical and expert knowledge based image transforms data augmentation. Comput. Vis. Pattern Recognit. (2017)
24. Igl, M., Ciosek, K., Li, Y., et al.: Generalization in reinforcement learning with selective noise injection and information bottleneck (2019)
25. Wang, S.H., Tang, C., Sun, J., et al.: Multiple sclerosis identification by 14-layer convolutional neural network with batch normalization, dropout, and stochastic pooling. Front. Neurosci. **12** (2018)
26. Singh, P., Yadav, A.K., Singh, K.: Color image encryption using affine transform in fractional Hartley domain. Optica Applicata **47** (2017)
27. Zhao, N., Yang, H.: Realizing speech to gesture conversion by keyword spotting. In: 2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP), pp. 1–5. IEEE (2016)
28. Li, Y., Chen, X., Zhang, X., et al.: A sign-component-based framework for Chinese sign language recognition using accelerometer and sEMG data. IEEE Trans. Biomed. Eng. **59**, 2695–2704 (2012)
29. Yang, H.-.D, Lee, S.-W.: Robust sign language recognition with hierarchical conditional random fields. In: 2010 20th International Conference on Pattern Recognition, pp. 2202–2205. IEEE (2010)
30. Anguita, D., Ghelardoni, L., Ghio, A., et al.: The 'K' in K-fold cross validation. In: ESANN, pp. 441–446 (2012)
31. Zhu, Z., Zhang, M., Jiang, X.: Fingerspelling identification for chinese sign language via wavelet entropy and kernel support vector machine. In: Satapathy, S., Zhang, YD., Bhateja, V., Majhi, R. (eds.) Intelligent Data Engineering and Analytics. AISC, vol. 1177, pp. 539–549. Springer, Singapore (2021). https://doi.org/10.1007/978-981-15-5679-1_52
32. Jiang, X., Zhang, Y.-D.: Chinese sign language fingerspelling via six-layer convolutional neural network with leaky rectified linear units for therapy and rehabilitation. J. Med. Imaging Health Informat. **9**, 2031–2090 (2019)

33. Jiang, X., Hu, B., Chandra Satapathy, S., et al.: Fingerspelling identification for Chinese sign language via AlexNet-based transfer learning and Adam optimizer. Sci. Progr. **2020**, 1–13 (2020)
34. Gao, Y., Zhu, R., Gao, R., Weng, Y., Jiang, X.: An optimized seven-layer convolutional neural network with data augmentation for classification of chinese fingerspelling sign language. In: Fu, W., Xu, Y., Wang, SH., Zhang, Y. (eds.) ICMTEL 2021. LNICST, Part II, vol. 388, pp. 21–42. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-82565-2_3