[1, 2]Yong Xu
[1] Manping Qin
[1] Cuiru Zhou
[*, 1] Jiansheng Peng
[2] Malathy Batumalay
[2] Choon Kit Chan

Research on Dynamic Gesture Recognition and Control System based on Machine Vision

Journal of Electrical Systems

Abstract: - Hand gesture recognition and control is a new type of human-computer interaction that can provide a more convenient and efficient operation mode by utilizing non-contact gesture recognition technology. This paper presents a lightweight dynamic gesture recognition method for intelligent office presentation control. First, we introduce the concept of hand gesture recognition and go over key gesture recognition technologies like classification. The structure, process, and evaluation index of the gesture recognition algorithm are described in detail using a convolutional neural network model. During the experiment's algorithm verification phase, we test and analyze the algorithm using the Python language, compilation environment, and data set. In the control experiment, we evaluated the system's ability to control the office application's start, play, next, previous, and exit functions. We achieve 96.3% accuracy on the test set. Experimental results show that the system can recognize a wide range of hand gestures and accurately control the presentation.

Keywords: Shunt Hand gesture recognition; convolutional neural network; human-computer interaction; Python; ShuffleNet.

I. INTRODUCTION

Hand gestures have the advantages of being intuitive, natural, and comfortable when compared to expression, movement, and other communication methods. As a result, gestures are the most common mode of communication other than language, and they are an important component of human-computer interaction, which is widely used in everyday life and work. Gesture, as a form of nonverbal communication, transmits information through specific actions and has a positive impact [1]. In some cases, if direct operation is not feasible, gestures can be used to operate, and the use of gesture recognition technology offers numerous benefits and potentials [2, 3].

Gesture recognition technology is widely used in fields such as Internet of Things [4], unmanned aerial vehicles [5], intelligent driving [6], medical assistance [7], and virtual reality [8]. Gesture recognition can help hearing-impaired people communicate in medical settings. Deaf and dumb people use gestures to convey their meaning, and non-signers can promote communication by recognizing and understanding. For example, gesture recognition systems in vehicles enable drivers to control entertainment and navigation functions through intuitive gestures, enhancing convenience and safety on the road. Gesture interaction technology enhances game interactivity and immersion by recognizing player gestures in real time and mapping them to the actions of game characters. Gesture recognition makes smart homes more user-friendly and intelligent. Residents can use specific gestures to control the lighting, temperature, television, and other devices.

In general, the use of gesture recognition technology in various fields can improve the user interaction experience by providing more convenient and intuitive operation methods. Currently, people primarily use tools such as data glove [9], Kinect camera [10], EMG signal [11], and sensor array [12, 13] to achieve gesture recognition. The use of special devices to collect gesture features is common to all of these methods. Additionally, machine vision technology [14, 15] employs image processing algorithms to extract intricate gesture features, enhancing the system's ability to interpret hand movements accurately. The former has advantages in recognition accuracy and anti-interference capability, whereas the latter has low equipment and environmental requirements as well as a broader range of applications, making it the first choice of the new generation of human-computer interaction.

Gesture recognition encompasses crucial stages such as gesture segmentation, tracking to monitor hand movements, feature extraction to identify key characteristics, classification to interpret gestures, and interaction for seamless communication with devices[14]. In recent years, the adoption of end-to-end neural networks has surged, with deep learning methods gaining popularity in gesture recognition[15-17]. These networks offer unique advantages such as seamless end-to-end learning and have revolutionized the field by enabling more

[1] School of Artificial Intelligence and Smart Manufacturing, Hechi University, Hechi 546300, China
[2] INTI International University, Nilai 71800, Malaysia
[*]Corresponding Author: Jiansheng Peng

efficient and accurate gesture recognition. Dynamic gesture recognition methods are classified into four categories: convolutional neural network (CNN) [18, 19], recurrent neural network (RNN) [19], attention mechanism [20], and hybrid neural network [21, 22]. However, the deep learning method has some drawbacks, including the need for a large amount of experimental data, high hardware requirements, and the creation of a large model size, which is inconvenient for subsequent embedding on mobile devices.

Currently, the continuous development of various network technologies has accelerated the pace of intelligent office upgrading and promoted the transition from touch interaction to gesture interaction. The evolution of intelligent offices empowers individuals to leverage cutting-edge technology and optimize information utilization, leading to enhanced office productivity, streamlined processes, and the realization of a more efficient, automated, and information-driven workspace.

This paper introduces a lightweight machine vision gesture recognition method that enables the implementation of remote control gestures for presentations, leveraging the capabilities of machine vision technology to enhance interaction and control during presentations. First, the basic process, concept, and key technology of the gesture recognition algorithm are examined, followed by an introduction to the recognition algorithm evaluation index. The images are then collected by OpenCV, the data are labeled by the LabelImg tool, the experiment's environment is built by Anaconda, and the code is executed on the Pycharm platform. Finally, the training, testing, and control of the remote control gesture recognition model for presentation, which can be used in smart offices, are completed.

II. METHODOLOGY

A. Design of Gesture recognition algorithm

Hand gestures can be classified as dynamic or static based on their state [2, 23], as well as two-dimensional planar hand gestures or three-dimensional hand gestures based on the recognition object [2]. Among them, 2D dynamic gesture recognition has the advantages of good information expression ability, small data scale, easy collection, and low processing platform requirements, making it the first choice for many practical application scenarios. As a result, this paper focuses on two-dimensional dynamic hand gesture recognition and control.

Figure 1 depicts the structure of the gesture recognition algorithm used in this paper, with the core contents being preprocessing, training, and testing. Real-time gesture images are first obtained, followed by image preprocessing, which includes image enhancement and sharpening, to improve image clarity [24]. The gesture feature values are then calculated to extract the contour, and the Cr component in the YCbCr color space is combined with the threshold segmentation algorithm for binarization, yielding the cutting image data. The image is then classified using a convolutional neural network, followed by training, testing, and analysis of the recognition results.

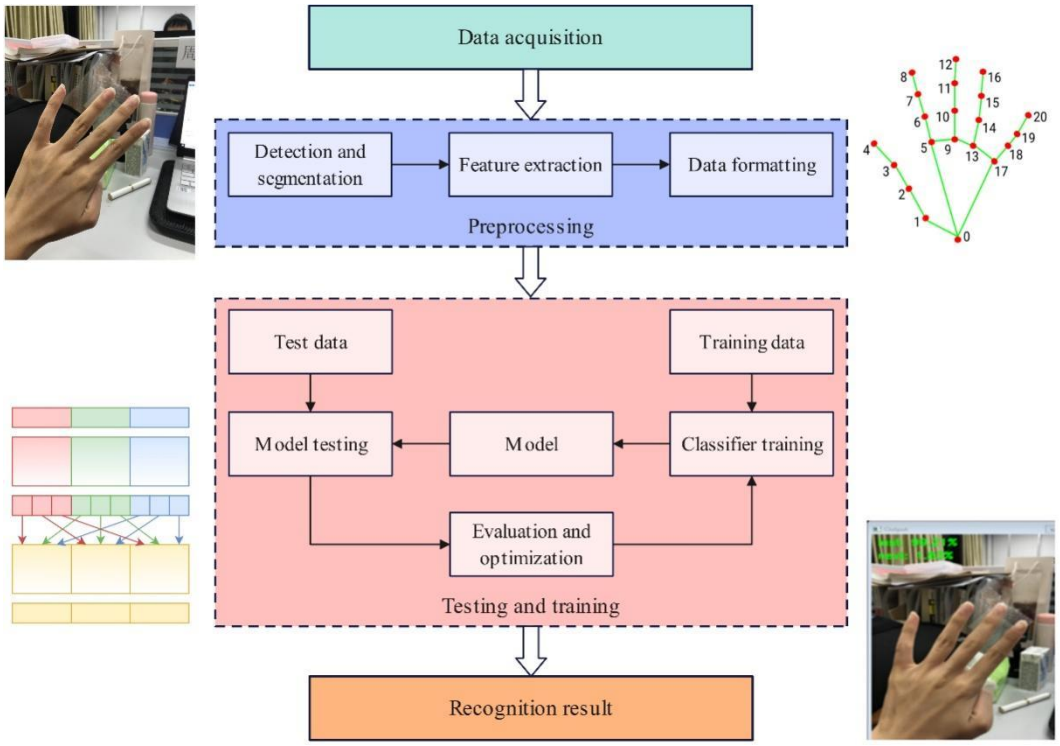

Figure 1. Structure of gesture recognition algorithm

B. Data Preprocessing

a. Image detection and segmentation

After capturing hand images with cameras or videos, image detection and segmentation are required. The purpose of detection and segmentation is to distinguish the gesture from the background in the image. To recognize dynamic gestures, hand tracking is required in addition to hand detection and segmentation. To achieve stable skin color segmentation, we used the RGB and YCbCr hybrid color space segmentation algorithm [25] with a pre-determined threshold. The YCbCr color space can be used to maximize the RGB image display. The RGB color space to YCbCr space conversion formula is:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & 0.081 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \quad (1)$$

The threshold for the hybrid color space segmentation algorithm is set as [25, 26]:

$$\begin{cases} R > G \wedge R > B \\ (G \geq B \wedge 5R - 12G + 7B \geq 0) \vee (G < B \wedge 5R + 7G - 12B \geq 0) \\ Cr \in (135,180) \wedge Cb \in (85,135) \wedge Y > 80 \end{cases} \quad (2)$$

b. Gesture feature extraction

Gesture feature extraction is a critical component of gesture recognition. Shape, motion, texture, color, and time-frequency features are among the most common vision-based gesture features [27, 28]. The selection of an appropriate feature extraction method is determined by the characteristics of the gestures, task requirements, and dataset characteristics. Feature selection algorithms can be used to assess the correlation between features and gesture categories and choose the most discriminative features to improve the accuracy and robustness of gesture recognition.

We use shape features to extract gesture features. Shape features describe gestures primarily based on their geometric characteristics. Google Research created MediaPipe, a multimedia machine learning model application framework that is open source [29]. MediaPipe can easily obtain coordinates for the gesture's key points. The gesture can be obtained simply by judging the distance between the joint points or the state of the drop. The points 0-20 in Figure 1 represent the finger coordinates we used to improve viewing gestures.

c. Data formatting

Formatting the data entails dimensionality compressing the three-channel image data after converting it into Numpy arrays in order for the classifier to accept the input data specifications. For batch processing in the convolutional neural network model, we use the Keras library. Store the image data for the corresponding category in the training and validation set paths, and label the folder with the class name. This enables the code to automatically retrieve class and sample data for each class, as well as extract class labels.

C. Classifier optimization and design

Classification is a natural language processing task based on the Machine Learning Algorithm (MLA). The classifier is chosen to combine the gesture features with the classifier, which is then trained using machine learning algorithms such as Support Vector Machine (SVM), deep learning, and so on, to finally recognize gestures. A classifier is any algorithm capable of carrying out a classification function, such as template matching, neural networks, probability and statistics models, and so on [30].

In recent years, the deep learning-based gesture recognition algorithm has emerged as one of the most popular and effective. Deep learning gesture recognition algorithms, unlike traditional ones, extract and classify gesture features using deep neural networks. CNN [18, 31], 3-Dimensional Convolutional Neural Networks (3DCNN) [32-34], and RNN [19] all have good sign language recognition effects, as well as the advantages of fast speed and a wide application range. In this paper, the ShuffleNet model of CNN is used to extract image features, and its fully connected layers serve as classification algorithms.

a. ShuffleNet model design

ShuffleNet, like MobileNet, is primarily intended for mobile devices. As an efficient and lightweight CNN model. The ShuffleNet model is built around two operations: pointwise group convolution and channle shuffle. These two steps allow the ShuffleNet model to significantly reduce its computational complexity while maintaining model performance.

Figure 2 shows the basic unit design of ShuffleNet. As illustrated in Figure 2(a), a basic residual unit with three layers is introduced: a 1 x 1 convolution is applied first, followed by a 3 x 3 Depthwise convolution (DWConv). The goal of implementing DWConv is to reduce the amount of computation. This 3x3 convolution, also known as the bottleneck layer, is followed by a 1 x 1 convolution and then a short-circuit connection to add the input directly to the output.

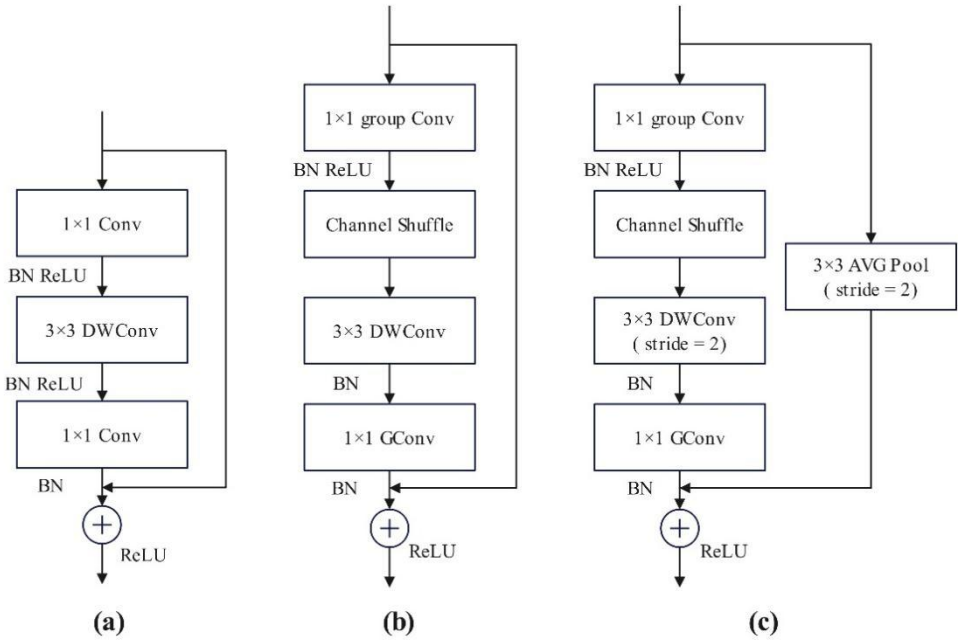

Figure 2. Model of ShuffleNet. (a) Residual structure. (b) Basic unit of network. (c) Basic unit of short circuit network

As shown in Figure 2(b), the following enhancements are made based on this residual structure: First, the standard 1 x 1 convolution is replaced with a 1 x 1 group convolution. We replace the previous operations with a channle shuffle after the first 1 x 1 convolution. This operation is only applied after the first 1 x 1 convolution, not after the 3 x 3 convolution.

Furthermore, unlike the other layers, no activation function is applied following the 3 x 3 DWConv operation. For the residual unit, the size mismatch between the input and output is typically addressed by mapping the input to the same size as the output using a 1 x 1 convolution. However, the ShuffleNet model adopts a different strategy, as shown in Figure 2 (c): 3 x 3 average pooling with step size 2 is applied to the original input, with a 3 x 3 convolution kernel size. This operation reduces the amount of calculation and parameter size by connecting the obtained feature map and the output. Table 1 illustrates the ShuffleNet network structure:

Table 1. Network structure of ShuffleNet

Items	Output	KSize	Stride	Repeat	Groups(g=1)
Input	128x128	-	-	-	3
Conv1	64x64	3x3	2	1	24
MaxPool	32x32	3x3			
Stage1	16x16	-	2	1	144
	16x16	-	1	3	144
Stage2	8x8	-	2	1	288
	8x8	-	1	7	288
Stage3	4x4	-	2	1	576
	4x4	-	1	3	576
GlobalPool	1x1	4x4	-	-	-
Softmax	-	-	-	-	24

To study the network employed, this research will use multiple hyperparameters, analyze the influence of different hyperparameters on the experimental findings, and determine the hyperparameter setting that yields the maximum recognition rate based on the experimental results. The learning rate is 0.04, the batch size is 200, the epoch is 50, the dropout is 0.5, the activation function is Softmax, and Adam is the optimizer.

b. Model comparison

We evaluate the convolutional network model based on three parameters: recognition accuracy, number of Parameters, and number of Floating Point Operations (FLOPs). Recognition accuracy is a parameter introduced in Section 2.4 that measures the effect of the model. It is defined as the ratio of the number of correct recognizations to the total number of recognizations.

The number of Parameters in a convolutional network model corresponds to its space complexity, which can be calculated as follows:

$$Parameters = (k_w \times k_h \times c_{in}) \times c_{out} + c_{out} + (n_{in} \times n_{out}) + n_{out} \quad (3)$$

In this equation, k_w represents the convolution kernel's width, k_h represents its height, c_{in} represents the number of channels in the current layer, c_{out} represents the number of output channels, n_{in} represents the number of nodes after stretching the input into a one-dimensional vector, and n_{out} represents the number of nodes in the layer's output. As can be seen, the number of fully connected layer parameters is only proportional to the number of input and output nodes.

FLOPs refer to the complexity of the time dimension, which is a parameter used to assess the complexity of an algorithm or model. It's calculated as:

$$FLOPs = [(k_w \times k_h \times c_{in}) \times c_{out} + c_{out}] \times H \times W + (n_{in} \times n_{out}) + n_{out} \quad (4)$$

The size of the output feature map of this layer is represented by $H \times W$. The convolution layer's feature maps are obtained by $(k_w \times k_h \times c_{in}) \times c_{out} + c_{out}$ convolution operations. It is clear that the computation amount of the convolutional layer is determined not only by the height width of the convolution kernel and the number of channels in the current layer's input and output, but also by the size of the input feature map.

We conducted comparative experiments using the VGG16, ResNet50 [35], MobileNet-V1 [36], and ShuffleNet-V1 models. The experimental data set consists of 500 pictures of various gestures in American Sign Language (ASL) [37], all of which are labeled. Table 2 shows the following results:

Table 2. Experimental comparison of various model networks

Model	Accuracy/%	Parameters/M	FLOPs/G
VGG16	84.3	136.3	14.962
ResNet50	93.6	25.7	4.081
MobileNet-V1	91.0	7.1	0.316
ShuffleNet-V1	92.2	4.2	0.115

The experimental results show that our improved ShuffleNet-V1 model has a recognition accuracy of 92.2%. ResNet50 outperforms other CNN models in terms of recognition accuracy, but the difference is subtle. The parameter number of the ShuffleNet-V1 model is 4.2, which is significantly lower than the traditional models VGG16 and ResNet50 with higher accuracy, and the effect is obvious. At the same time, the FLOPs of the ShuffleNet-V1 model are 0.115G, which is less than that of the traditional model, and the effect is clear. As a result, we propose ShuffleNet, a lightweight CNN, to address the problem of low model efficiency while also optimizing the traditional model's storage and prediction speed.

c. Classifier implementation

Data processing dataPrep and the cnnModel classifier complete the CNN classifier code implementation. To begin, define the image pixel format in the data processing function, then import the training, test, and validation sets of data, as well as the pre-trained model's parameters and the ShuffleNet model itself. In dataPrep, the training and validation sets were pre-trained on the ShuffleNet deep neural network, respectively. The Keras library can automatically classify and detect the sample type and amount of data; simply store different categories in the corresponding folder when importing the data. We then place the pre-trained features from the training and validation sets in the bottleneck Numpy array.

Finally, the local CNN classifier was trained. Similarly, the image pixel format was defined, the training, test, and validation sets of data were imported, the classifier model parameters were set, the sample type label and data amount were classified and detected, the classifier's number of fully connected layers and dropout

parameters were set, and the pre-training results were imported for training. Save the trained parameters locally. The training results are then visualized, and a classification test on random images is run.

D. Algorithm evaluation metrics

There are complete evaluation indicators in the field of object detection that can also be used to evaluate the results of gesture recognition. The Confusion Matrix is typically used to assess the precision and accuracy of detection, whereas the Frame Per Second (FPS) of the algorithm is typically used to assess detection speed. Precision (P), Average Precision (AP), and Accuracy(ACC) are some of the most commonly used evaluation indicators [38].

A Confusion Matrix is a common tool for evaluating the performance of classification models, as shown in Table 3. It is a two-dimensional table that compares the difference between model predictions and actual labels. The rows in the Confusion Matrix represent true labels, while the columns represent the model's predictions. To compare the results visually, we define four key terms: True Positives (TP) are the number of positive examples that the model correctly predicts as positive. False Negatives (FN) are the number of examples that the model incorrectly classifies as negative. False Positives (FP) are the number of examples that the model incorrectly classifies as positive. True Negatives (TN) are the number of negative examples that the model correctly predicts as being negative.

Table 3. Confusion Matrix

Items	Predicting positive examples	Predicting negative examples
Actual positive examples	TP	FN
Actual negative examples	FP	TN

Analyzing the confusion matrix allows you to understand how well the model predicts across different categories and adjust the model to improve performance. In this experiment, three indicators were used: ACC, P, and AP. These indicators are typically appropriate for recognition algorithms based on bounding boxes and are frequently used to assess algorithm performance. The object detection algorithm's performance can be evaluated by taking into account detection accuracy, localization accuracy, and so on. According to the definition of a confusion matrix, the selected indicators are:

Precision is the fraction of true positives among all positive examples; it is the classifier's ability to predict whether a positive example will be correct. P is calculated as follows:

$$P = \frac{TP}{TP+FP} \tag{5}$$

AP is used to evaluate the algorithm's overall performance in terms of detection accuracy and recall across multiple categories. The area under the PR curve can be used to calculate the average precision for each class. The AP ranges from 0 to 1, with higher values indicating that the object detection algorithm performs better across different classes.

The ACC represents the proportion of correctly predicted samples as well as the total number of samples classified. ACC measures an algorithm's accuracy, or a classifier's ability to state that it is correct. The ACC is computed as follows:

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \tag{6}$$

There are also Recallrate, mean Average Precision (mAP), and F1 score, which measure the performance of object detection algorithms and classification models, respectively. The experimental section of this paper will use the above indicators to evaluate the hand gesture recognition algorithm.

III. EXPERIMENT AND RESULTS

This experiment was carried out on an Intel i7-1165G7 CPU and an NVIDIA T500 (GPU, 4095MB), with each training using a single card and all of the hyperparameters set to the optimal values listed in Section 2.3. The programming language is Python3.6, and the development environment is PyCharm Community 2020.3. Python libraries used in the experiment include h5py 2.8.0, imageio 2.9.0, keras 2.3.1, numpy 1.15.0, opencv 3.4.2, pandas 1.0.5, scikit-image 0.23.2, scikit-learn 0.23.2, tensorflow 1.4.0, and so on.

A. Dataset construction

In this paper, OpenCV is used to create a data set that is divided into a training and test set, and multi-label image classification is implemented using Keras. This design is divided into three parts: first, collect a large enough dataset. Second, the network structure is established using Keras, the network model ShuffleNet under CNN is implemented, and the training data set is created. Finally, the trained model was used to classify the test data.

Our image dataset contains 5 basic gestures that represent program presentation commands: the FOUR, FIVE, SIX, GOOD, and OK gestures. The self-created gesture data set was used to collect gesture language images from the author of this paper for the construction and testing of the gesture recognition algorithm. The training set includes 75 images of each of the five gesture types, and Figure 3 depicts a subset of the collected data.

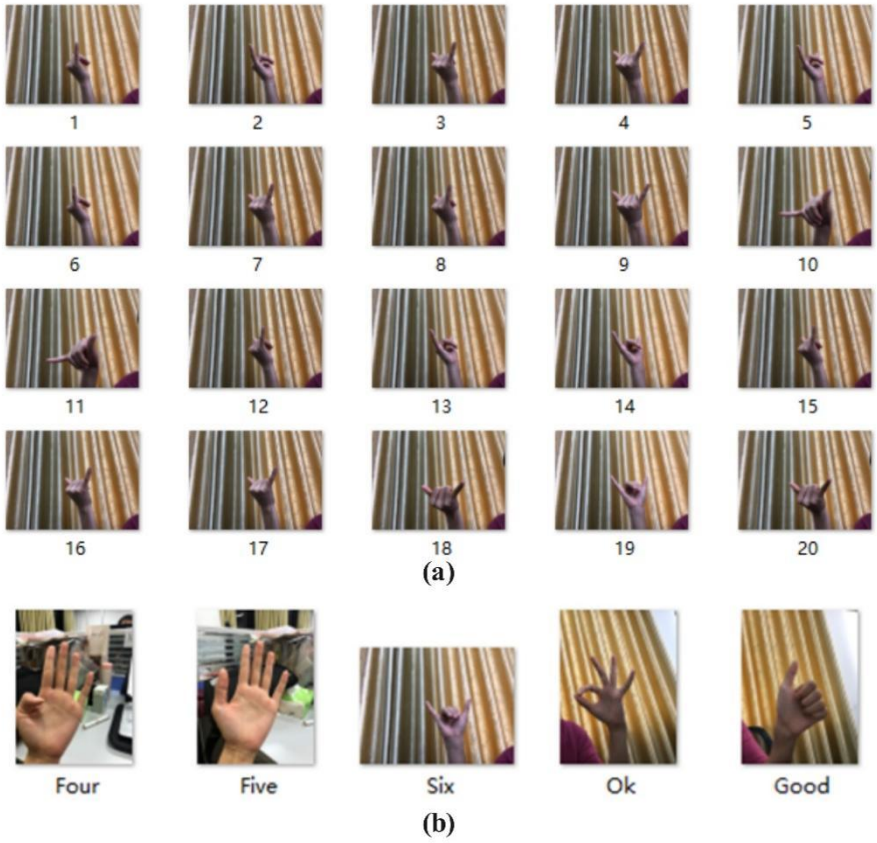

Figure 3. Image of hand gesture. (a) Part of the SIX gesture diagram. (b) Diagrams of basic hand gesture

Read the image through OpenCV, then perform image preprocessing using the method described in Section 2.2, and finally directly input the machine vision image into CNN. After preprocessing, the images' gesture features are obvious, and compression will not harm the features, so we reduce the training images to 128x128 pixels.

B. Defining classifier

ShuffleNet from the Keras library is used as the CNN model in this experiment. Prior to input data, pre-training is performed on the ImageNet dataset. Large neural network training requires a lot of memory and time, so when pre-training the network, the data only goes through the convolutional and pooling layers, not the fully connected layer. Feature weight values are extracted at bottleneck features to fine-tune small neural networks used in practice.

The pre-trained feature values were then imported into the training set, and the Keras sequential model was used to create a one-dimensional input layer. The fully connected layers are then classified using three LeakyRelu activation functions of dimensions 100, 50, and 5. A dropout layer is added to the three fully connected layers for regularization, with proportions of 50% and 30%, respectively, which means that 50% and 30% of nodes are randomly dropped each time to avoid overfitting. The experimental model's epoch is set to 7, and the initial batch_size is 50, which is automatically optimized based on training.

C. Training

Figure 4 depicts the effect of the CNN model we built using Kreas trained on the training set, with a total of 50 rounds of actual training and 12 rounds of training each. In CNN model training, 5% to 10% of the data is designated as the validation group. Because the total number of datasets in this experiment is small, cross validation is performed using 10% of each.

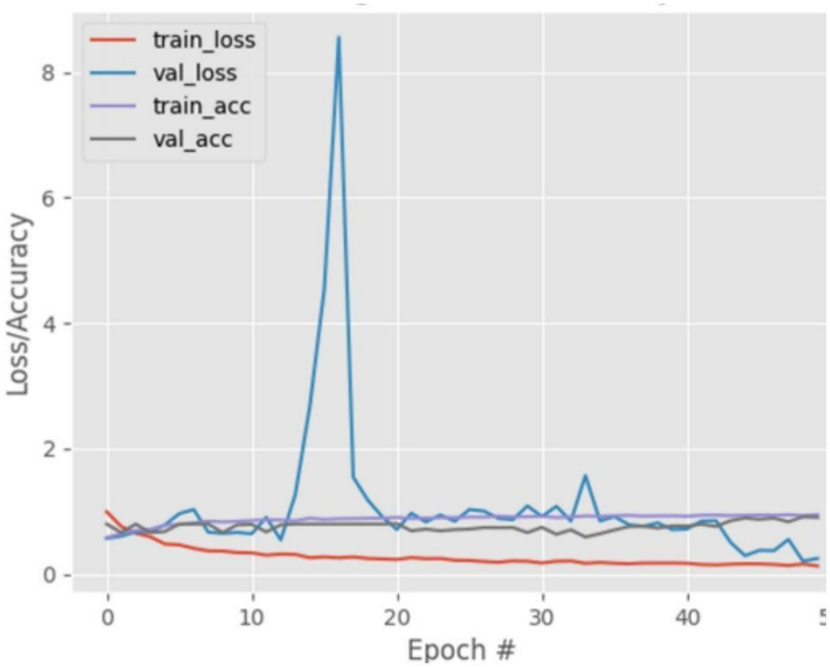

Figure 4. ACC and loss of CNN in the training set

Following each batch operation, the CNN classifier calculates loss and accuracy and returns information such as operation time. Figure 4 shows that the CNN model's average ACC in the test set is 96.3%, with a cross-entropy loss rate of only 0.05. At the same time, the blue broken line represents the test set loss rate, which varies greatly when the training number ranges between 10 and 20. This is because the training data set used in the design of this paper is insufficiently large, but the number of training times is relatively large, resulting in overfitting. However, as the number of training sessions increases, the loss rate gradually decreases to the average range, indicating that the experimental results are correct. Overfitting can be solved by increasing or decreasing the number of training sessions.

D. Testing and analysis

Real-time hand gesture testing involves calling the pre-trained CNN model stored on the computer for online gesture judgment. The camera captures the finger joints and uses Python and OpenCV to display the identified results on the window interface. In this paper, only the first two results with the highest recognition rate are presented. If you want to know the recognition rate of different gesture actions, you can change the program. We tested five different types of gestures online, and Figure 5 depicts the gesture recognition effect when the background was yellow curtains with no face interference or occlusion. Table 4 shows the precision of each gesture action test.

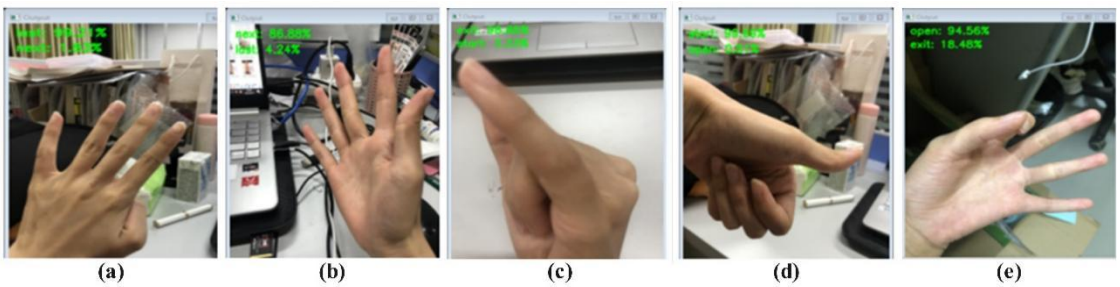

Figure 5. Images of real-time gestures recognition. (a) FOUR gesture. (b) FIVE gesture. (c) SIX gesture. (d) GOOD gesture. (e) OK gesture

Table 4. The Precision of the five gesture movements

Hand gesture name	Function definition	Number of test samples	P/%
FOUR	Play previous page	50	98.2
FIVE	Play next page	50	90.6
SIX	END	50	98.1
GOOD	Launching application	50	98.7
OK	Start playing	50	95.9

In the course of the experiments, it was discovered that even in a complex environment, the network model can accurately judge the action state of the gesture, with the precision of each action recognition being 90% or higher, but individual cases are not excluded. For example, during testing, we discovered that the gesture for projecting the slideshow OK from the current position was misinterpreted as GOOD.

This is due to the common practice of using unprocessed online images, with the caveat that other processing may change the image's resolution. To ensure the test's authenticity, unprocessed images are used, and the experimental results have a low error rate, almost negligible, indicating that the experiment is correct. However, if it is a dataset gathered using other cameras with beauty features. To improve the recognition rate, data augmentation techniques such as rotation, scaling, dilation, and translation can be applied to the image. As a result, the experiment can demonstrate that our Shufflenet CNN model is highly accurate and capable of completing gesture recognition tasks.

IV. GESTURE RECOGNITION AND CONTROL SYSTEM

Currently, the traditional human-computer interaction method is used in the office. Although the introduction of wireless mice and keyboards has greatly increased people's convenience, they are clearly not intuitive to use [39]. As a result, it is necessary to create a new human-computer interaction mode that allows for intuitive control of office software without the use of any tools. The goal of this paper's human-computer interaction based on gesture recognition is to recognize gesture actions using computer vision without requiring contact with the entity to complete the interactive operation.

A. System flow

We used the Python GUI library's Tkinter module to create a gesture recognition and presentation control system. The flow of the proposed system is depicted in Figure 6. To begin the program, configure the code to turn on your computer or an external camera. The camera captures and analyzes the dynamic gesture before using the pre-trained model to recognize it. When a hand is detected rising or the distance between the finger joints changes, the gesture action is recognized and the appropriate gesture operation is executed. When the gesture action does not match the gesture that defines the operation in the program, it is recaptured. When the presentation has finished playing, the system program will also terminate.

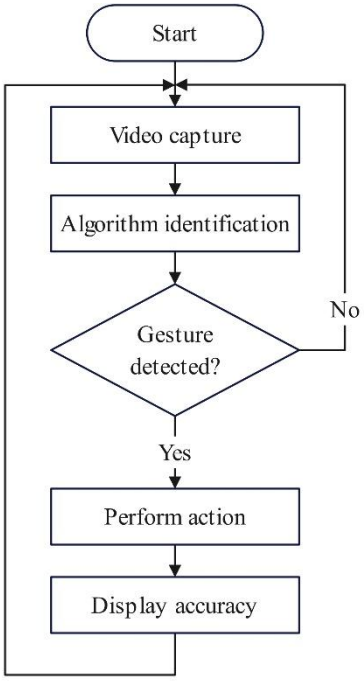

Figure 6. Flow diagram of gesture recognition control

B. Key code mapping

In order to complete the various operations on the presentation without using a keyboard or mouse. We are mapping the gesture state to the computer keyboard, so when the camera detects a gesture, it corresponds to pressing a key on the computer keyboard. For example, let's define the OK gesture as pressing the Shift and F5 keys simultaneously on a computer keyboard. You can control the show's presentation by holding down the Shift and F5 keys while pressing the shortcut key corresponding to the presentation of the "start from the current page" option. Similarly, other gesture actions can be used to complete presentation control using this operation; the function definition is shown in Table 3. To complete the page turning task, the control program uses the exit key Esc (Vk code 27), the F5 key (Vk code 116), the Shift key (Vk code 160), and the arrow keys UP (Vk code 38) and DOWN.

C. System testing

a. Launching application

When the system recognizes the corresponding gesture to launch the application software, it is used as an action instruction to press the appropriate key code. For example, if the camera detects the dynamic gesture state as GOOD, which is only the thumb raised and the other fingers down, the corresponding application will be launched. Each finger has its own one-dimensional array element, such as finger[0] for the thumb and finger[1] for the index finger. 0 and 1 indicate whether the finger is falling or rising; for example, finger[2]==1 indicates that the middle finger is rising, while finger[2]==0 indicates that the middle finger is falling.

This system uses the WPS application for testing, but it can also use other office software; simply change the path of the application in the os.startfile() function. The startfile(path,[operation]) function starts the application. Figure 7(a) shows that when only the thumb is captured, the WPS application starts and the frame rate is displayed in the top left corner of the camera capture workspace.

b. Start playing

When the captured hand gesture state is OK, the middle, ring, and little fingers are raised while the other fingers are lowered. At this time, it is equivalent to pressing the Shift key on the keyboard, then pressing the shortcut key F5 after a 0.4-second delay, and finally releasing the Shift and F5 keys after 0.4 seconds. The presentation can be started from the current position, and the computer screen will show an icon to do so. The plt.imshow(img) function completes the icon display operation. To do this, use mpimg.imread(image_path) to obtain the icon's absolute path.Imread can only read jpg and png images. The keybd_event(bVk, bScan, dwFlags, dwExtraInfo) function controls the presentation's page-turning behavior.

The workspace where the camera captures the gesture is hidden after entering the projection presentation, so it is not visible in the result figure, as shown in Figure 7(b). The camera captures the middle, ring, and pinky fingers rising, and a prompt appears above the screen to begin playing. It should be noted that once the icon workspace appears, the camera workspace will be unable to continue capturing the gesture. At this point, the Thread() function must be used to set the display time of the icon workspace. Allow the icon to display for a set period of time before going to sleep so that the camera can continue to capture the gesture.

c. Play previous page

When the FOUR gesture state is captured, i.e., the thumb is down and the other fingers are up, it is equivalent to pressing the up key on the keyboard, which is released after 0.4 seconds. The previous slide can be played, and the computer screen shows the previous page icon. The results are depicted in Figure 7(c).

d. Play next page

When the gesture state is captured as FIVE, that is, with all fingers raised. This is equivalent to pressing down on the keyboard and then releasing it after 0.4 seconds. The next slide can be played, and the next page icon will appear on the computer screen, as illustrated in Figure 7(d).

e. END

When the camera captures the SIX gesture, only the thumb and pinky fingers are raised, while the other fingers are down. This is equivalent to pressing the Esc key and then releasing it after 0.3 seconds. The slide can be completed from its current position, and the computer screen will show the end projection icon, as shown in Figure 7(e).

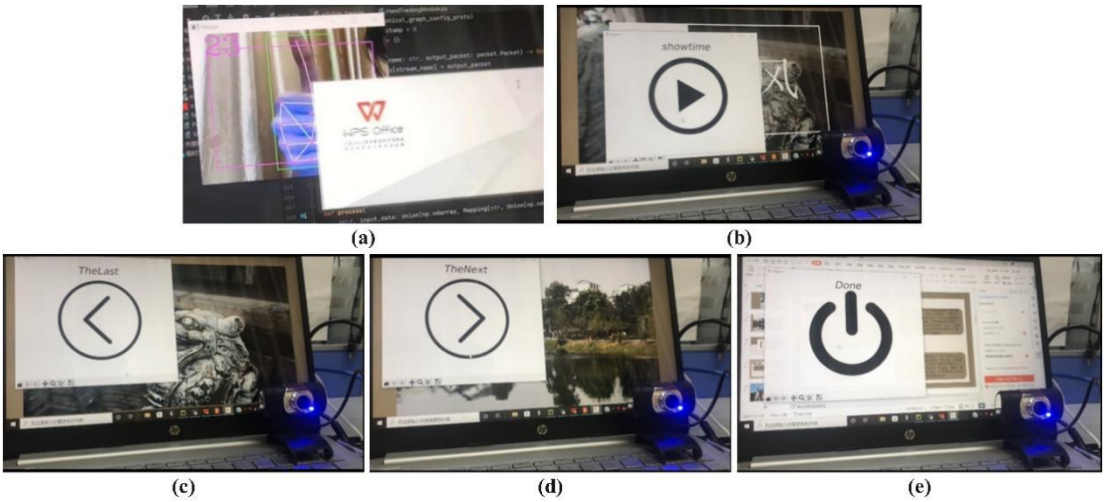

Figure 7. System test diagram of gesture recognition and control. (a) Launching WPS. (b) Start playing. (c) Play previous page. (d) Play next page. (e) END

In short, we use camera recognition to extract the distance and depth of each joint point in the gesture and save the operator's gesture action. The trained CNN model was incorporated into the system to complete gesture recognition. A gesture recognition and presentation control system is created using Python programming. The system function has been verified following the five types of hand gesture recognition and presentation control test experiments described above.

V. CONCLUSIONS

In this paper, we design a dynamic hand gesture recognition algorithm based on a lightweight CNN model using machine vision algorithm architecture. The network model has been pre-trained on large-scale data sets in order to obtain gesture feature values and classify them. Simultaneously, this paper provides a detailed introduction to the algorithm's implementation in Python. Experiments show that the Shufflenet algorithm model is strong in generalization, has high recognition accuracy, and can successfully complete the gesture recognition task. In addition, we combined the hand gesture recognition algorithm to create a hand gesture recognition control system, which allows us to intuitively control the presentation to perform the corresponding action.

The era of artificial intelligence for gesture recognition is approaching, and it aims to provide a more convenient and faster way of life and work for human society. Hand gesture recognition intelligent office systems should fully utilize information resources in order to achieve office information, automation, and paperless.

ACKNOWLEDGEMENT

This research was financially supported by First-class Discipline Construction Project of Hechi University, Guangxi Colleges and Universities Key Laboratory of AI and Information Processing (Hechi University), Education Department of Guangxi Zhuang Autonomous Region.

Funding Statement: This research was funded by the Natural Science Foundation of Guangxi Province under Grant No: 2023GXNSFAA026025, the Project of Ministry of Education supply and demand docking employment education under Grant No: 20230106490, Special Research Project of Hechi University under Grant No: 2022YLXK003, Research Project of Hechi University under Grant No: 2023XJYB009.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

REFERENCES

[1] Kendon, A. (1997). Gesture. Annual review of anthropology, 26(1), 109-128.

[2] Oudah, M., Al-Naji, A., Chahl, J. (2020). Hand gesture recognition based on computer vision: a review of techniques. journal of Imaging, 6(8), 73.

[3] Sharma, H.K., Choudhury, T. Applications of Hand Gesture Recognition. Challenges and Applications for Hand Gesture Recognition. IGI Global; 2022. p. 194-207.

[4] Nair, R., Singh, D.K., Yadav, S., et al. (2020). Hand Gesture Recognition system for physically challenged people using IoT. 2020 6th international conference on advanced computing and communication systems (ICACCS): IEEE, 671-675.

[5] Patrona, F., Mademlis, I., Pitas, I. (2021). An overview of hand gesture languages for autonomous UAV handling. 2021 Aerial Robotic Systems Physically Interacting with the Environment (AIRPHARO), 1-7.

[6] Graichen, L., Graichen, M., Krems, J.F. (2022). Effects of gesture-based interaction on driving behavior: a driving simulator study using the projection-based vehicle-in-the-loop. Human factors, 64(2), 324-342.

[7] Van Amsterdam, B., Funke, I., Edwards, E., et al. (2022). Gesture recognition in robotic surgery with multimodal attention. IEEE Transactions on Medical Imaging, 41(7), 1677-1687.

[8] Dube, T.J., Johnson, K., Arif, A.S. (2022). Shapeshifter: Gesture Typing in Virtual Reality with a Force-based Digital Thimble. CHI Conference on Human Factors in Computing Systems Extended Abstracts1-9.

[9] Dong, Y., Liu, J., Yan, W. (2021). Dynamic hand gesture recognition based on signals from specialized data glove and deep learning algorithms. IEEE Transactions on Instrumentation and Measurement, 70, 1-14.

[10] Li, Y. (2012). Hand gesture recognition using Kinect. 2012 IEEE International Conference on Computer Science and Automation Engineering: IEEE, 196-199.

[11] Ahsan, M.R., Ibrahimy, M.I., Khalifa, O.O. (2011). Electromygraphy (EMG) signal based hand gesture recognition using artificial neural network (ANN). 2011 4th international conference on mechatronics (ICOM): IEEE, 1-6.

[12] Lien, J., Gillian, N., Karagozler, M.E., et al. (2016). Soli: Ubiquitous gesture sensing with millimeter wave radar. ACM Transactions on Graphics (TOG), 35(4), 1-19.

[13] Berman, S., Stern, H. (2011). Sensors for gesture recognition systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(3), 277-290.

[14] Mohamed, N., Mustafa, M.B., Jomhari, N. (2021). A review of the hand gesture recognition system: Current progress and future directions. IEEE Access, 9, 157422-157436.

[15] Neverova, N., Wolf, C., Taylor, G.W., et al. (2015). Multi-scale deep learning for gesture detection and localization. Computer Vision-ECCV 2014 Workshops: Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part I 13: Springer, 474-490.

[16] Mujahid, A., Awan, M.J., Yasin, A., et al. (2021). Real-time hand gesture recognition based on deep learning YOLOv3 model. Applied Sciences, 11(9), 4164.

[17] Devineau, G., Moutarde, F., Xi, W., et al. (2018). Deep learning for hand gesture recognition on skeletal data. 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018): IEEE, 106-113.

[18] Mohanty, A., Rambhatla, S.S., Sahay, R.R. (2017). Deep gesture: static hand gesture recognition using CNN. Proceedings of International Conference on Computer Vision and Image Processing: CVIP 2016, Volume 2: Springer, 449-461.

[19] Prakash, K.B., Eluri, R.K., Naidu, N.B., et al. (2020). Accurate hand gesture recognition using CNN and RNN approaches. International Journal, 9(3).

[20] Lv, X., Dai, C., Liu, H., et al. (2023). Gesture recognition based on sEMG using multi-attention mechanism for remote control. Neural Computing and Applications, 35(19), 13839-13849.

[21] Uke, S.N., Zade, A.V. (2023). An enhanced artificial neural network for hand gesture recognition using multi-modal features. Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization, 1-12.

[22] Akintola, K., Emmanuel, J. (2020). Static hand gesture recognition using multi-layer neural network classifier on hybrid of features. Am J Intell Syst, 10, 1-7.

[23] Gao, Q., Chen, Y., Ju, Z., et al. (2021). Dynamic hand gesture recognition based on 3D hand pose estimation for human–robot interaction. IEEE Sensors Journal, 22(18), 17421-17430.

[24] Al-Hammadi, M., Muhammad, G., Abdul, W., et al. (2020). Deep learning-based approach for sign language gesture recognition with efficient hand gesture representation. Ieee Access, 8, 192527-192542.
[25] Eshaq, R.M.A., Hu, E., Li, M., et al. (2020). Separation between coal and gangue based on infrared radiation and visual extraction of the YCbCr color space. Ieee Access, 8, 55204-55220.
[26] Kumar, A., Raja, L., Dadheech, P., et al. (2020). A hybrid cluster technique for improving the efficiency of colour image segmentation. World Review of Entrepreneurship, Management and Sustainable Development, 16(6), 665-679.
[27] Camurri, A., Moeslund, T. (2010). Visual gesture recognition. Musical Gestures-Sound, Movement, and Meaning.
[28] He, J., Zhang, C., He, X., et al. (2020). Visual recognition of traffic police gestures with convolutional pose machine and handcrafted features. Neurocomputing, 390, 248-259.
[29] Singh, A.K., Kumbhare, V.A., Arthi, K. (2021). Real-time human pose detection and recognition using mediapipe. International Conference on Soft Computing and Signal Processing: Springer, 145-154.
[30] Yang, Y., Duan, F., Ren, J., et al. (2020). Performance comparison of gesture recognition system based on different classifiers. IEEE Transactions on Cognitive and Developmental Systems, 13(1), 141-150.
[31] De Smedt, Q., Wannous, H., Vandeborre, J.-P. (2016). Skeleton-based dynamic hand gesture recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops1-9.
[32] Tran, D., Bourdev, L., Fergus, R., et al. (2015). Learning spatiotemporal features with 3d convolutional networks. Proceedings of the IEEE international conference on computer vision4489-4497.
[33] Hu, Z., Hu, Y., Liu, J., et al. (2018). 3D separable convolutional neural network for dynamic hand gesture recognition. Neurocomputing, 318, 151-161.
[34] Ozcan, T., Basturk, A. (2019). Transfer learning-based convolutional neural networks with heuristic optimization for hand gesture recognition. Neural Computing and Applications, 31, 8955-8970.
[35] Raksha, A., Rajasekaran, R.K., Francis, P., et al. (2021). Home Automation through Hand Gestures Using ResNet50 and 3D-CNN. Digital Presentation and Preservation of Cultural and Scientific Heritage, 11, 215-226.
[36] Suharto, E., Widodo, A., Sarwoko, E. (2020). The use of mobilenet v1 for identifying various types of freshwater fish. Journal of Physics: Conference Series: IOP Publishing, 012105.
[37] Crespo, R.G., Verdú, E., Khari, M., et al. (2019). Gesture recognition of RGB and RGB-D static images using convolutional neural networks. IJIMAI, 5(7), 22-27.
[38] Ahmad, S., Ansari, S.U., Haider, U., et al. (2022). Confusion matrix-based modularity induction into pretrained CNN. Multimedia Tools and Applications, 81(16), 23311-23337.
[39] Guo, L., Lu, Z., Yao, L. (2021). Human-machine interaction sensing technology based on hand gesture recognition: A review. IEEE Transactions on Human-Machine Systems, 51(4), 300-309.