IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 8, 2018, accepted September 28, 2018, date of publication October 5, 2018, date of current version October 31, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2874022

Human Action Recognition Algorithm Based on Adaptive Initialization of Deep Learning Model Parameters and Support Vector Machine

FENG-PING AN[1,2]

[1]School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China
[2]School of Physics and Electronic Electrical Engineering, Huaiyin Normal University, Huai'an 223300, China

e-mail: anfengping@163.com.

This work was supported by the National Natural Science Foundation of China under Grant 61701188.

ABSTRACT Action recognition is a trending topic and key research direction in computer vision, machine learning, artificial intelligence, and other fields. This research seeks to identify human action in image and video data. Its research results have been widely used in the fields of safety monitoring, disability monitoring, understanding multimedia content, human–computer interaction, virtual reality, and so on. However, the existing traditional human action recognition technology has many limitations in practical application, such as low accuracy and weak adaptive ability. Although the action recognition based on deep learning can self-learn and improve the action recognition accuracy, there are many difficulties in training the deep neural network model, such as gradient disappearance, gradient explosion, and overfitting. Therefore, this paper will reduce the abovementioned difficulties in deep neural network model training from the perspective of deep neural network model parameter initialization and then propose a model parameter initialization method based on the multilayer maxout network activation function to solve the difficulties in deep neural network model training. Then, on this basis, a method of learning the temporal and spatial characteristics of human action based on the deep neural network model is proposed. First, the method detects and tracks the human action and uses the restricted Boltzmann machine (RBM) to encode the temporal and spatial features of various parts of the human body. Second, the temporal and spatial feature codes of various parts of the human body are integrated into a global temporal and spatial feature representation method of the action video through a RBM neural network. Finally, the trained SVM classifiers are used to recognize human action. Experiments show that the human action recognition method proposed in this paper not only has high recognition accuracy but also has great adaptability. Thus, this method extracts temporal and spatial features from the shape feature sequences of various parts of the human body, thus opening up a new way to extract human action features and solving the problem of human action recognition in complex scenes. Its proposal provides an exploratory technical method and approach for self-adaptive recognition of human action. It also gives directional enlightenment to the development and improvement of self-adaptive human action methods.

INDEX TERMS Human action recognition, pose estimation, initialization method, spatial-temporal features, deep learning.

I. INTRODUCTION

The extraction and learning of action characteristics is the most important step in the process of action recognition. In recent years, action recognition technology has been widely used in daily life and industrial fields [1]–[3]. According to different action feature extraction methods, action recognition algorithms can be divided into the following three categories: action recognition methods based on traditional artificial design features [4]–[6], action recognition methods based on deep learning features [7]–[9], and hybrid feature extraction action recognition methods [10], [11]. The hybrid feature-based action recognition method herein is the fusion of artificial feature extraction and deep learning action feature extraction, that is, the hybrid feature extraction is performed

through a combination of artificial design features and deep learning strategies.

The first type of research results that use artificial action features to extract and identify are more common [3]–[5]. Therefore, a large number of global features [12] and local features [5] are also described in the area of action recognition. These artificial design features are sophisticated, and they provide very good action recognition effects through the characterization of human action. However, in recent years, there have been no breakthroughs and advances in the action recognition of actions through artificial design features [4], [5].

Deep learning technology has achieved great success in speech recognition, image processing and other aspects, and the second type of action recognition methods based on deep learning has also received more attention. Many scholars have used deep learning techniques to obtain a good recognition effect in action recognition. For example, Reference [13] uses the techniques of stacking and convolution to learn the unchanging spatial-temporal features directly from the video data and uses this feature to obtain a very good action recognition effect. Reference [14] proposed a method of action recognition based on convolutional neural network architecture. This method learns a potential feature of an image sequence from continuous image pairs. Then, an experiment verifies the reliability of the method to identify the action. Reference [15] uses a three-dimensional convolutional neural network to recognize the human action in a video in an end-to-end manner. Without relying entirely on design features, it gains an overwhelming action recognition effect. However, in the three-dimensional video, the computational complexity of the deep learning methods that are very convenient in image processing becomes very large, and many operations become extremely complicated. In the case of convolutional operations, the computational complexity of the video increases exponentially. Although convolutional operations based on deep learning have made great progress in development and application, there has also been a problem of falling into local optimum and slow training. Thus, the training process of the deep neural network model is a non-convex optimization process, and the solution of the non-convex optimization algorithm (such as stochastic gradient descent) depends on the initialization of model parameters [16], [17]. The initialization of deep neural network model parameters is a very important part of the model training process. It directly affects the distribution of each hidden layer at the initial stage of the neural network model. Proper model parameter initialization can ensure that the state of each hidden layer node obeys the same distribution and stability of gradient propagation. It can increase the convergence speed of non-convex optimization processes and avoid the non-convex optimization process from becoming trapped into a local optimal solution. Inappropriate model initialization of parameters will cause the problem of gradient disappearance or gradient explosion when the gradient spreads due to the large difference in hidden layer distribution. It will lead to slow convergence and easily fall into the local optimal solution [18], [19]. In practical applications, the model parameter initialization method is related to the structure of the model, the nonlinear activation function, and other factors. When a deep neural network model with different structures and different nonlinear activation functions is encountered, the model parameter initialization method needs to be redesigned. This process requires much experimentation and theoretical derivation. It not only makes the model training process cumbersome and time-consuming, but it is also easy to improperly initialize the model parameters, causing the model training to be difficult to converge. At present, most of the research work on model parameter initialization is based on the assumption that each hidden layer obeys a similar distribution and carries out logical analysis and theoretical derivation of both forward and back propagation of the network. The nature of the nonlinear activation function at each level of the network plays an important role. For example, the Xavier initialization method is based on the sigmoid activation function, and the MSRA initialization method is based on the ReLU activation function [18]–[20]. Since the nature of the Maxout and MMN activation functions is not the same as the traditional activation function, it causes the above two model parameter initialization methods to be inapplicable to the Maxout and MMN activation functions.

The third category is an action recognition method based on hybrid feature extraction [10], [11]. There are few research results regarding this type of method at present, as it uses deep-learning technology to learn the features of artificial design, which are extracted to obtain more abstract features, and then engages in action recognition. For example, Reference [10] uses deep learning to identify action based on the human skeleton and joints. Reference [11] uses the DPM (deformable part model) to detect human parts or targets and then uses deep learning methods to identify the action through the detected position information of various parts of the body or target. Given the effectiveness of this approach in action recognition, this type of hybrid strategy has also received more attention.

Through actual monitoring, it can be determined that human action consists of a series of ordered sequences of limb movements. The movement of various parts of the body forms a sequence of changes in the shape of the various parts of the human body in the time dimension. The shape change sequence has an important influence on action recognition. We can learn this type of shape change feature by using the video angle to capture the movement laws of human action. In view of the fact that shape features have achieved good results in many action recognition algorithms, Boltzmann machines are also limited in the distribution of learning data [21]–[25]. This paper attempts to use the video feature learning action recognition algorithm. This method uses a deep neural network to extract the shape changes of various parts of the human body in the video, learn the characteristics of the action, and then identify the specific action through the features. This paper proposes an adaptive model parameter

initialization method based on the MMN linear activation function, which overcomes the problem of gradient disappearance and gradient explosion during training to avoid the influence of deep neural network model parameter initialization on the effect of deep learning. It can provide a more theoretical guarantee for effectively training the network model based on the MMN activation function. First, according to the nature of the MMN activation function, the forward propagation process of the neural network model is deduced, and a sufficient condition for the initial model parameters to be satisfied under the same distribution condition is obtained. Second, according to the loss of the network forward propagation calculation, the back-propagation algorithm deduces the back-propagation process of the model, and the gradients of all hidden layers in the back-propagation (BP) process satisfy the same distribution, and the initial model parameters must satisfy sufficient conditions [26]. Finally, in the process of forward propagation and backward gradient propagation, the sufficient conditions for the above two basic satisfactions are merged to propose a method for parameter initialization of the deep learning adaptive model. Based on the above description and considering the importance of the initialization of the parameters of the deep neural network model and the characteristics of the traditional feature extraction methods, this paper proposes a temporal and spatial feature human action recognition algorithm based on the parameters of the deep learning model.

Section II of this paper will mainly describe the space-time feature-learning algorithm based on deep learning. Section III systematically explains the initialization method of the deep learning model parameters proposed in this paper. Section IV introduces the action recognition classifier based on Support Vector Machine. Section V analyzes the action recognition algorithm proposed in this paper and compares it with the mainstream recognition algorithm. Finally, the full text is summarized and discussed.

II. SPACE-TIME FEATURE LEARNING BASED ON DEEP LEARNING

This section describes how to use deep learning theory to study the temporal and spatial characteristics of human behavior. The feature learning process is divided into four steps. (1) This method uses pedestrian detection and tracking algorithms to identify the action features; the measured action video is converted into action tracking sequences [23]–[26] (Action Tracks), and then feature learning is performed through action tracking sequences. The specific architecture is shown in Fig. 1. The architecture consists of a multilayered neural network whose basic unit is a restricted Boltzmann machine (RBM). (2) The action tracking sequence is divided into a plurality of video blocks as shown in Fig. 1, and each frame of image in each video block is segmented to obtain a shape feature (Block Shape Features). This method does not learn features directly from the pixel information of the video block but rather learns more abstract features from the extracted video block shape features.

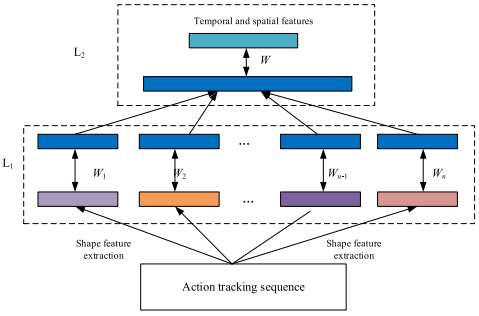


FIGURE 1. Space-time feature learning framework.

(3) Feature extraction from the shape feature sequence of the video block occurs using a plurality of restricted Boltzmann machine-neural network layers. (4) The output of the multi-restricted Boltzmann machine-neural network layer is spliced and input into the second neural network. The output of the second layer network is the space-time characteristics learned. The purpose of setting up the second layer neural network is to reduce the dimension of the output of the multi-restricted Boltzmann machine-neural network layer, thereby reducing the amount of computation and improving the computational efficiency. The following describes the learning process of the space-time features in detail.

A. VIDEO SEQUENCE ACTION TRACKING

The human body undergoes action changes during specific action movements. The concrete manifestation thereof is that the position of the human body in the detected video frames will change; the posture of the human body will appear with corresponding changes. This article uses the target detection and tracking algorithm to automatically detect and track the action of the human body to ensure that the human body's specific action is always in the visual focus and turn the detection and tracking results into action tracking sequences (action track). In the initial frame of the action video, the person in action is detected using a mainstream pedestrian detection algorithm. After the active human body is detected, the method uses a tracking algorithm to locate the active human body in a subsequent frame to place the actor in the visual focus in a subsequent video frame. The positioning of the human body in action will seriously affect the accuracy of the subsequent action recognition. Therefore, it is very important to perform the detection of the specific action successfully.

However, when an actor performs a variety of different actions, his limbs perform a variety of specific movements. Based on the results of pedestrian detection and tracking, it is very difficult to determine the bounding box that contains the actor's body parts. Therefore, this paper uses a larger bounding box to locate the actor to limit the actor's limbs and trunk to the bounding box under various movement postures. Similar to the action recognition method, the proposed method also optimizes the bounding box based on pedestrian

detection and tracking results. The optimized rectangular bounding box is centered on the center axis of the bounding box obtained by a pedestrian detection algorithm or tracking algorithm, and its width is proportional to the height of the bounding box. Finally, the action tracking sequence is formed by normalizing the actor's positioning results in the same scale.

To facilitate subsequent processing, the length of the action tracking sequence is set to a fixed length T. If the length of the initial action tracking sequence is greater than T, the redundant video frames are discarded; otherwise, the zero-padding method is used to extend the action tracking sequence to T frames. In this paper, for the action category ci, the action tracking sequence of all the training videos is denoted as T_{c_i}, and the action tracking sequence of other action types is denoted as T_{b_i}.

B. VIDEO BLOCK SHAPE FEATURES

Each action tracking sequence is segmented into $s_w \times s_h$ video blocks from the beginning of the video. As described above, the frame length of each video block is set to a fixed value T. The segmented video block is denoted by $F_{j,}, j \in J$, where $J = \{1, 2, \cdots, s_w \times s_h\}$ corresponds to the spatial position of the video block. Since processing a video block sequence using a three-dimensional convolution method results in a very large number of computations and is time-consuming, the proposed method represents a video block sequence as a video block shape feature. Then, the deep neural network is used to obtain more abstract spatial-temporal features from these low-level features. Divide each frame of the video block $F_j, j \in J$, F_k^j $(k = 1, 2, \cdots, T)$ into $N_w \times N_h$ grid cells and compute each grid cell in N_d directions in the gradient direction histogram (HOG). The splicing vector of the gradient direction histogram of all the grid cells of each frame image represents the shape characteristics of the image frame. Therefore, the dimension of the shape feature of each image frame is $M_w \times M_h \times M_w$. In reference [21], this shape feature is represented as a feature vector $(m_{k1}^j, m_{k2}^j, \cdots, m_{kn}^j)$, where $n = M_w \times M_h \times M_w$. $s_{kl}^j, l = 1, 2, \cdots, n$ indicates the first component of the shape feature of the image frame F_k^j. For each video block of the action tracking sequence, the shape features of each video frame are extracted and spliced into a long vector. This feature vector represents a shape feature called a video block. The shape features of the video block of the action tracking sequence. The first row of the graph is an image sequence of a video block of an action tracking sequence, and the second action is its corresponding shape feature of the video block.

In the action recognition algorithm, the active person's posture is very important information. This paper normalizes the shape features of each frame of the action tracking sequence. That is, the shape features of the human pose in each frame of the action tracking sequence are normalized. According to previous experience, the $L2$ norm is very effective for splicing image description features. Thus, a frame of the image of the action tracking sequence is represented as $F_k^1, F_k^2, \cdots, F_k^{s_w \times s_h}, k = 1, 2, \cdots, n$. The normalization of the shape features of each frame of the image in the action tracking sequence is as follows:

$$q_{kl}^j = \frac{m_{kl}^j}{\left(\sum_{j=1}^{s_w \times s_h} \sum_{r=1}^{m} \left| m_{kr}^j \right|^2\right)^{\frac{1}{2}}} \tag{1}$$

Where $1 \le l \le m$, q_{kl}^j is the normalization of the shape feature vector, and the component m_{kl}^j of the shape feature vector corresponds to the normalized component value. In summary, the shape feature of each image frame in the video block of the action tracking sequence is described as $D_k^j = \left(q_{k1}^j, q_{k2}^j, \cdots, q_{km}^j\right)$, where $j \in J, 1 \le k \le T$. Then, the video block shape feature of the video block Bj can be represented as $\left(D_1^j, D_2^j, \cdots, D_T^j\right)$. The dimension of this feature is $T \times M_w \times M_h \times M_w$. $q_{kl}^j \in [0, 1]$, so, eigenvector $\left(D_1^j, D_2^j, \cdots, D_T^j\right)$ can be used as an RBM input to train the two-layer neural network architecture designed in this paper.

C. MULTI-RBM NEURAL NETWORK LAYER

RBM is an undirected graphical model, which is a special type of Markov random field. The RBM is a network architecture that contains two layers of neurons. The two layers of neurons are input layer neurons and hidden layer neurons. There is no connection between the same layer of neurons in the network, and the input layer and the neurons in the hidden layer are connected in a fully connected manner. This type of neural network model was first proposed in [27]. Subsequently, Freund discussed the learning algorithm of the network in [28]. An effective learning algorithm for training Boltzmann machines was proposed in [29]. Freund *et al.* [28] noted that RBMs can learn any discrete distribution when there are enough hidden neurons. As shown in Fig. 1, the first layer of a neural network consists of multiple RBMs. The proposed method uses multiple RBM neural network layers to describe the characteristic distribution of the action. The structure of the multiple RBM neural network layers is shown in Fig. 2. As previously mentioned, the video block of the action tracking sequence is represented herein as a video block shape feature. For each action category, the video block shape features of all training samples of the action type are used to train the RBMs of the multiple RBM neural network layers. Each RBM is trained using the video block shape feature of the corresponding spatial location. Correspondingly, the multi-RBM neural network layer contains $s_w \times s_h$ needed to train the RBM.

In Fig. 2, the output layer of each RBM contains K neurons, and the value of K directly affects the distribution of the characteristics of each type of action learned. Therefore, this method specifically analyzes the influence of the value of K on the experimental results. For each RBM neural network layer, limit each Boltzmann machine $(j = 1, \cdots, s_w \times s_h)$.

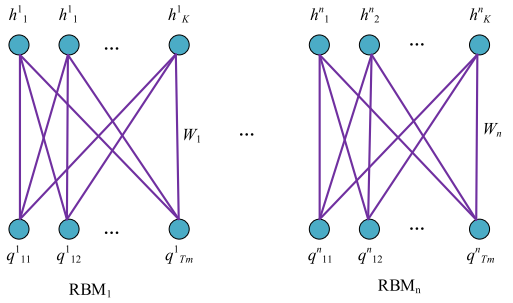


FIGURE 2. Multiple RBM neural network layer structure.

It trains the shape features of all video blocks with the corresponding spatial position j as input. The input video block has the shape feature $Q^j = \left(q^j_{11}, q^j_{12}, \cdots, q^j_{Tm}\right)^T$ and the corresponding RBM_{j} output $Hj = H^j = \left(h^j_1, h^j_2, \cdots, h^j_K\right)^T$. For the restriction of the Boltzmann machine RBM_{j}, the state energy of its neurons $\{Q_{\mathrm{j}}, \}$ is defined as

$$\begin{aligned} &E\left(Q^j, H^j; \theta^j\right) \\ &= -\left(Q^j\right)^T W^j H^j - \left(b^j\right)^T Q^j - \left(a^j\right)^T H^j \\ &= -\sum_{l=1}^{T\times m}\sum_{k=1}^{K} W^j_{lk} Q^j_l H^j_k - \sum_{l=1}^{T\times m} b^j_l Q^j_l - \sum_{k=1}^{K} a^j_k H^j_k \end{aligned} \quad (2)$$

Where $\theta^j = \{W^{\mathrm{j}}, \}$ is a parameter of RBM_{j}, and W_{j} represents a symmetric correlation matrix between the input neuron and the output neuron. It refers to the connection weight between the input layer and the output layer. b_{j} and a_{j} are deviation vectors, both of which are column vectors. Each RBM parameter is learned through the contrastive divergence (CD) algorithm [29]. For aRBM_{j}, the joint distribution between the input neurons and the output neurons is

$$P\left(Q^j, H^j; \theta^j\right) = \frac{1}{Z\left(\theta^j\right)} \exp\left(-E\left(Q^j, H^j; \theta^j\right)\right) \quad (3)$$

$$Z\left(\theta^j\right) = \sum_{Q^j}\sum_{H^j} \exp\left(-E\left(Q^j, H^j; \theta^j\right)\right) \quad (4)$$

Among them, (θ_{j}) is a partition function, and the conditional probability distribution can be easily derived from Formula 3, specifically

$$p\left(H^j_k = 1|Q^j\right) = g\left(\sum_l W^j_{lk} Q^j_l + a^j_k\right) \quad (5)$$

$$p\left(Q^j_L = 1|H^j\right) = g\left(\sum_k W^j_{lk} H^j_k + b^j_l\right) \quad (6)$$

Among them, $(x) = 1/(1 + exp(-x))$ is a logic function. The contrastive divergence [29] method is used to learn R's parameter set. This paper separately trains each RBM of the first layer network (multi-RBM neural network layer) of the used neural network architecture. The set of network parameters that this multi-RBM neural network layer learns for each action class is denoted by $\theta = (\theta^1, \cdots, \theta^{\mathrm{n}})$.

D. TEMPORAL AND SPATIAL FEATURES

This paper trains a two-layer neural network for each action category. As shown in Fig. 1, the second layer of the neural network is a single RBM. The layer network is designed to reduce the dimension of the output of the first neural network. The parameters of this layer network are the same as each RBM of the first layer neural network, denoted as $(W, a,)$. For trained neural networks of different action categories, the input of the same action video will cause the network of each action category to output various types of feature vectors.

The output of the trained two-layer neural network is the learned space-time characteristics. Unlike those features based on deep learning methods that learn directly from the original pixel values of the video, the spatiotemporal characteristics of this article are learned from the shape features of the video block. This feature is an abstract high-level feature learned from traditional low-level features that can be more robust to characterize the action. The space-time characteristics of learning are denoted as $H = (h_1, h_2, \ldots, h_{\mathrm{S}})$. Among them, the value of S is the set based on experience. In the experiments in this paper, we set $S = 16 \times \mathrm{s_w} \times \mathrm{s_h} \times K$.

III. MODEL PARAMETER ADAPTIVE INITIALIZATION METHOD BASED ON THE MAXOUT ACTIVATION FUNCTION

A. MODEL PARAMETER INITIALIZATION METHOD

Before training the deep neural network model, we must first initialize the weight parameters of the model; otherwise, the model cannot be optimized according to the initial state. Proper parameter initialization can avoid the problem of gradient disappearance or gradient explosion in the back-propagation process and accelerate the convergence of model training. Inappropriate initialization of model parameters will lead to the optimization of the model parameters into a locally optimal solution, which results in the model being unable to be fully trained and unable to exert the powerful feature extraction capabilities and feature representation capabilities of the deep neural network model. Currently, many scholars have conducted studies on the parameter initialization method of the deep neural network model and have achieved certain research results [30]–[32] that can quickly and effectively select appropriate initial values for different depth neural network model parameters. They not only overcome the gradient disappearance and gradient explosion problems that often appear in the deep neural network model training process but also accelerate the training convergence rate of the deep neural network model.

The traditional neural network model parameter initialization method initializes model parameters randomly from a Gaussian distribution [33]. All hidden layer parameters have the same Gaussian distribution. That is, all parameters in the

model obey the following Gaussian distribution:

$$W \sim N\left(\mu, \sigma^2\right) \tag{7}$$

Although this model parameter initialization method is simple to implement, it lacks a theoretical basis. The two hyperparameters of the Gaussian distribution need to be determined through multiple experiments. The determination of these two hyperparameters is very time-consuming and does not guarantee the validity of the initialization results of the model parameters. When using deeper, more complex deep neural network models (for example, more than 8 neural network models for convolutional layers), using this model parameter initialization method does not guarantee that the model has good performance. The Visual Geometry Group (VGG) of Oxford University performed several experiments on the VGG model (13 convolutional layers and 3 fully connected layers) and showed that the deep convolutional neural network model using this method is very difficult to converge [33]. This problem also limits the application and development of deep convolutional neural network models in image classification tasks.

With the deepening of the research on the optimization process of the deep neural network model, many parameter initialization methods for deep neural network models are emerging. These model parameter initialization methods can be roughly divided into two categories. One is based on a pre-training model parameter initialization method. For example, the method of initializing the model parameters through the unsupervised pre-training method proposed by Hinton *et al.* [34] in 2006, the deep belief network (DBN) [35], deep Boltzmann machine (DBM), DBM) [36] achieved significant results on equal depth neural network models. This model parameter initialization method builds an unsupervised objective function layer by layer, using an unsupervised model such as a restricted Boltzmann machine [37] and an autoencoder (AE) [38]. Layer-by-layer training can obtain the feature representation of the input features in the new space. This completes the initialization process for each level of the deep neural network model parameters. Whether it is a model parameter initialization method based on unsupervised pre-training or a model parameter initialization method based on supervised layer-by-layer greedy pre-training, the pre-training process requires extra computational overhead. It affects the training efficiency of the deep neural network model to some extent. The other is a model parameter initialization method based on neural network model training and parameter optimization. For example, to quickly and efficiently initialize the parameters of the deep neural network model, many researchers started with the nonlinear activation function and the back-propagation process. Bradley et al. found that as the gradient propagates from the output layer to the input layer, the back-propagation makes the gradient decrease, resulting in low-level parameters that cannot be effectively trained. By analyzing the gradient of the sigmoid activation function in the definition domain, we find that the gradient of the sigmoid activation function is relatively large in the interval $[-4, 4]$, which can accelerate the training speed of the neural network model. Glorot's research work in 2010 [39] deeply analyzed the gradient changes in different hidden layers during training. Based on the assumption that every layer of neuron nodes should obey the same distribution, the Xavier model parameter initialization method was proposed. Although the Xavier model parameter initialization method provides a quick and effective reference and guidance for parameter initialization of the deep neural network model, the Xavier model parameter initialization method is based on the traditional neural network model inference of the sigmoid activation function and the tanh activation function. It does not apply to neural network models that use ReLU activation functions and their variants. In 2015, He *et al.* [19] assumed that the state of each neuron node should obey the same distribution and proposed a parameter initialization method for the MSRA model that is suitable for the ReLU activation function and its variants. Due to the wide application of the ReLU activation function, the initialization of MSRA model parameters solves the problem of initialization for many deep neural network model parameters. However, since the inference process of the MSRA model parameter initialization method is based on the ReLU activation function, this model parameter initialization method is only applicable to the deep neural network model using the ReLU activation function and its deformation.

B. MODEL PARAMETER ADAPTIVE INITIALIZATION METHOD BASED ON THE MAXOUT ACTIVATION FUNCTION

As mentioned earlier, the MMN activation function cannot only alleviate the gradient disappearance and gradient explosion problems in the back-propagation process, but it can also increase the feature extraction ability and feature representation capability of the neural network model. It is jointly optimized with other parameters in the deep neural network model. It will further improve the image classification accuracy of the deep neural network model. Although the scale of the deep neural network model using the MMN activation function has increased, a reasonable model parameter initialization method can accelerate the convergence of the model training. The existing model parameter initialization method is based on the traditional activation function because the theoretical derivation assumptions are not applicable to the deep neural network model using the MMN activation function or the Maxout activation function. This paper is based on the assumption that the state of each neuron node obeys the same distribution to initialize the parameters of the neural network model using the MMN activation function or the Maxout activation function more quickly and efficiently. A neural network model parameter adaptive initialization method based on the MMN activation function and the Maxout activation function is proposed.

Because the MMN activation function is a multi-layer Maxout network, the model parameter initialization method based on the Maxout activation function is also effective for the deep neural network model using the MMN activation function. Therefore, the Maxout activation function is used as an example for the theoretical derivation of model parameter initialization. The forward propagation process and back-propagation process of the deep convolutional neural network model will be analyzed separately to ensure that the state of each neuron node obeys the same distribution.

1) FORWARD PROPAGATION PROCESS

To ensure the derivation of the forward propagation process of the deep convolutional neural network model, first, the following assumption is made: All the input vectors s and the parameter vectors W are independent and obey the same distribution; the initialization distribution of the parameter vector W is symmetrical about the zero point; the offset b of each layer is always equal to zero.

Here, l denotes the l-th hidden layer of the deep convolutional neural network model, and the response of the l-th convolution layer in the deep convolutional neural network model is:

$$z_l = x_l^T W_l + b_l \tag{8}$$

Among them, xl∈Rd, xl is the original input vector or the state vector of the previous hidden layer. After the original input vector is preprocessed, the mean value is zero, then:

$$d = p^2 c \tag{9}$$

Where d represents the number of all input nodes connected to one neuron node, p represents the size of the convolution kernel (convolution kernels are square), and c represents the number of input channels. The output of each neuron node that passes through the Maxout activation function can be calculated by Formula (10), so the variance of z_l can be obtained, as shown in (11):

$$f(x) = \max\left(w_1 x + b_1, w_2 x + b_2, \cdots, w_n x + b_n\right) \tag{10}$$

Where n represents the number of linear functions in the combination. When $w_1 = 1$ and b_1, w_2, b_2, …, w_n, b_n are equal to zero, the Maxout activation function is equivalent to the ReLU activation function. The local linearity of the Maxout activation function alleviates the problem of gradient disappearance, but at the same time, it introduces additional parameters, which means that it takes more computing resources and storage resources during the training process.

$$Var[z_l] = d_l Var[W_l x_l] \tag{11}$$

Since the weight W_l of the l hidden layer obeys the zero-mean Gaussian distribution, the weight W_l and the state vector x_l are independent of each other. Therefore,

$$Var[z_l] = d_l Var[W_l] E\left[x_l^2\right] \tag{12}$$

Among them, $E\left[x_l^2\right]$ is the expectation of x_l^2. Only the Maxout activation function consisting of two linear functions is considered here to simplify the presentation,.

$$x_l = h_{l-1}(x_{l-1}) = \max\left(z_{l-1,1}, z_{l-1,2}\right) \tag{13}$$

Since the offset b_{l-1} is always equal to zero, the mean value of the weight W_l is also equal to zero, so both $z_{l-1,1}$ and $z_{l-1,2}$ are symmetrical about the zero point and the mean is equal to zero.

x_l is defined as shown in (14) to establish the connection between the expectation $E\left[x_l^2\right]$ and the variance $Var[z_{l-1}]$.

$$x_l = \frac{z_{l-1,1} + z_{l-1,2} + \left|z_{l-1,1} - z_{l-1,2}\right|}{2} \tag{14}$$

Substituting (14) into calculation expectation $E\left[x_l^2\right]$, we get:

$$\begin{aligned} &E\left[x_l^2\right] \\ &= \frac{1}{4} E\left[\left(z_{l-1,1} + z_{l-1,2} + \left|z_{l-1,1} - z_{l-1,2}\right|\right)^2\right] \\ &= \frac{1}{2} E\left[z_{l-1,1}^2 + z_{l-1,2}^2 + \left(z_{l-1,1} + z_{l-1,2}\right)\left|z_{l-1,1} - z_{l-1,2}\right|\right] \\ &= \frac{1}{2}\left(\begin{array}{l} E\left[z_{l-1,1}^2\right] + E\left[z_{l-1,2}^2\right] \\ + \left(E\left[z_{l-1,1}\right] + E\left[z_{l-1,2}\right]\right) E\left[\left|z_{l-1,1} - z_{l-1,2}\right|\right] \end{array}\right) \\ &= \frac{1}{2}\left(Var\left[z_{l-1,1}\right] + Var\left[z_{l-1,2}\right]\right) \end{aligned} \tag{15}$$

Since $z_{l-1,1}$ and $z_{l-1,2}$ obey the same distribution, the variance of $z_{l-1,1}$ can be defined as shown in (16):

$$Var[z_{l-1}] = Var\left[z_{l-1,1}\right] = Var\left[z_{l-1,2}\right] \tag{16}$$

Substituting (16) into (15) obtains:

$$E\left[x_l^2\right] = Var[z_{l-1}] \tag{17}$$

By substituting (16) into (12), the relationship between $Var[z_l]$ and $Var[z_{l-1}]$ can be obtained:

$$Var[z_l] = d_l Var[W_l] Var[z_{l-1}] \tag{18}$$

When there are L hidden layers in the deep convolutional neural network model, the relationship between the variance $Var[z_l]$ of the first hidden layer state and the variance $Var[z_L]$ of the last hidden layer state can be expressed as:

$$Var[z_L] = \left(\prod_{l=2}^{L} d_l Var[W_l]\right) Var[z_l] \tag{19}$$

The state of each neuron node is ensured to obey the same distribution to reduce the internal covariate shift in each hidden layer in the neural network model, that:

$$Var[z_L] = Var[z_1] \tag{20}$$

The initialization of the neural network model parameters needs to satisfy the sufficient conditions, as shown in (21):

$$d_l Var[W_l] = 1, \forall l \tag{21}$$

When $l = 1$, the above sufficient condition (21) is true since there is no activation function directly acting on the input vector.

In summary, the hypothesis is that each hidden layer node state follows the same distribution based on the neural network model. The model parameter initialization method proposed in this paper requires that each hidden layer parameter W_l of the deep convolutional neural network model satisfies the Gaussian distribution as shown in (22).

$$W_l \sim N\left(0, \frac{1}{\hat{d}_l}\right) \tag{22}$$

2) BACK-PROPAGATION PROCESS

In the back-propagation process of the deep convolutional neural network model, it is necessary to pay attention to the gradient obtained by each convolutional parameter, as shown in (23):

$$\Delta x_l = W_l \overset{\Delta}{\Delta} h_l \tag{23}$$

Here, Δx_l and Δh_l denote gradients $\frac{\partial Loss}{\partial x_l}$ and $\frac{\partial Loss}{\partial h_l}$, respectively. Loss is the loss function of the neural network model. Δx_l is the $c \times l$ dimensional vector, and Δh_l is the $\hat{d} \times 1$ quantity, where $\hat{d} = p^2 e$ and e is the number of convolution filters. W and $\hat{W}_l$ can be transformed into each other by changing the dimension because $\hat{W}_l$ is the $c \times \hat{d}$ matrix. Similar to forward propagation, the relevant assumptions still need to be made here: Δh_l and W (or $\hat{W}_l$) are independent of each other; W (or Δh_l) follows the initialization distribution with respect to 0 symmetry; for all l, $E[\Delta x_l] = 0$.

Here, we still consider only the Maxout activation function a=0, to get:

$$\Delta z_{l,k} = f'\left(z_{l,k}\right) \Delta x_{l+1}, k \in \{1, 2\} \tag{24}$$

Among them, $f'\left(z_{l,k}\right) = 1$ and $f'\left(z_{l,k}\right) = 0$ each has half the probability of occurrence. Since $f'\left(z_{l,k}\right) = 1$ and Δx_{l+1} are independent of each other, they are satisfied for any $k \in \{1,2\}$.

$$E[\Delta h_l] = E\left[\Delta z_{l,k}\right] \tag{25}$$

Simultaneously,

$$E\left[(\Delta h_l)^2\right] = Var[\Delta h_l] = \frac{1}{2} Var[\Delta x_{l+1}] \tag{26}$$

Therefore, the variance of the gradient Δx_l is:

$$Var[\Delta x_l] = \frac{1}{2} \hat{d}_l Var[W_l] Var[\Delta x_{l+1}] \tag{27}$$

The relationship between $Var[\Delta x_l]$ and $Var[\Delta x_{l+1}]$ can be established by (27). When there are L hidden layers in the deep convolutional neural network model, the relationship between $Var[\Delta x_2]$ and $Var[\Delta x_{L+1}]$ can be deduced as follows:

$$Var[\Delta x_2] = Var[\Delta x_{L+1}] \left(\prod_{l=2}^{L} \frac{1}{2} \hat{d}_2 Var[W_l]\right) \tag{28}$$

To enable the gradient to propagate back to the hidden layer in front smoothly, sufficient conditions for the initialization of the network model parameters need to be satisfied.

$$\frac{1}{2} \hat{d}_l Var[W_l] = 1, \quad \forall l \in [2, L] \tag{29}$$

When the first hidden layer is initialized, formula (29) still applies because the first layer has no activation function directly acting on the input vector.

In summary, the initialization of the neural network model parameter W needs to follow the Gaussian distribution as shown in (30):

$$W_l \sim N\left(0, \frac{2}{\hat{d}_l}\right) \tag{30}$$

After the previous theoretical derivation, the initialization methods based on the forward propagation process and the reverse propagation process are obtained, but the two cannot be satisfied at the same time. To this end, here is an eclectic approach to translating the above issues into the following optimization problems:

$$\min_{\tau_l} (\tau_l - d_l)^2 + \left(\tau_l - \frac{1}{2} \hat{d}_l\right)^2 \tag{31}$$

Among them,

$$W_l \sim N\left(0, \frac{1}{\tau_l}\right) \tag{32}$$

Based on two sufficient conditions, the optimal solution gives a sufficient condition for trade-off, as shown in (33):

$$W_l \sim N\left(0, \frac{4}{2d_l + \hat{d}_l}\right) \tag{33}$$

Because the MMN activation function is a multi-layer Maxout network, the model parameter initialization method based on the Maxout activation function is also applicable to the deep convolutional neural network model using the MMN activation function.

3) VERIFICATION AND ANALYSIS

In order to verify the validity of the model parameter adaptive initialization method proposed in this paper, relevant experiments were carried out on the CIFAR-10 and ImageNet datasets respectively. The model used was a deep convolutional neural network model (four convolutional layers), and the loss function was calculated by a logistic loss layer connected after the average downsampling layer. All activation functions in the model used the activation function proposed in this paper. In order to prevent over-fitting of the model, the Dropout method was used after each downsampling layer [40]. That is, in the training phase, each batch of data was trained using a sub-model of a randomly selected convolutional neural network model, and all the sub-models shared all the parameters, with a Dropout probability of 50%. The image classification error rate of the model was the result of averaging in 5 experiments, and the model parameters were randomly initialized each time.

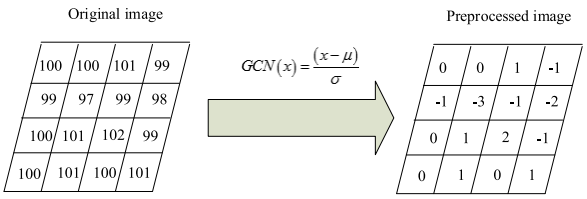


FIGURE 3. Global contrast normalization schematic.

(i) Data preprocessing

Before training, the training data was first preprocessed [41], including Global Contrast Normalization (GCN) and ZCA whitening. The GCN process is shown in Fig.3. ZCA whitening is defined as shown in (34).

$$X_{ZCA} = US^{-\frac{1}{2}}U^T X \tag{34}$$

In the formula (34), X is the data matrix, U is the eigenvector matrix of the data X covariance matrix, U^T is the transpose of the matrix U, and S is the eigenvalue matrix of the data X covariance matrix.

All models used the Xavier initialization method to initialize the parameters [39] and the initialization of the model parameters W obeys the uniform distribution U, which satisfies the requirements of formula (35).

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}, \right]\frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}} \tag{35}$$

In the formula, n_j is the number of input nodes, and n_{j+1} is the number of output nodes.

(ii) CIFAR-10 and ImageNet datasets verification and analysis

On the CIFAR-10 dataset, the model parameter adaptive initialization method proposed in this paper was compared with the Xavier model parameter initialization method. The training data was first processed by GCN and ZCA whitening. The deep convolutional neural network model in the experiment used the model parameter initialization method proposed in this paper and the Xavier model parameter initialization method respectively. The performance comparison of different parameter initialization methods is shown in Table 1.

TABLE 1. Test error rates of different initialization methods on the CIFAR-10 dataset.

Method type	Test error rates
Xavier model parameter initialization method	7.25±0.09%
The proposed model parameter adaptive initialization method proposed	6.21±0.07%

Using the model parameter adaptive initialization method proposed in this paper, the image classification performance of the deep convolutional neural network model using the MMN activation function was improved. The reason is that the proposed adaptive initialization method can ensure that the node states of different hidden layers obey the same distribution, effectively avoiding the problem of gradient disappearance or gradient explosion in model training.

In order to further verify the generalization performance of the model parameter adaptive initialization method proposed in this paper on the big dataset, a comparison experiment of different model parameter initialization methods was also carried out on the ImageNet data set. Since the ImageNet image dataset is a 1000-class image problem, a deep convolutional neural network model with more layers and convolution kernels was used in the experiment.

During the experiment, the deep convolutional neural network model iterated a total of 500,000 times, the initial learning rate was set to 0.1, and the learning rate decreased by 10 times after 15,000 iterations.

The performance comparison between the model parameter adaptive initialization method proposed in this paper and the Xavier model parameter initialization method on the ImageNet image classification data set is shown in Table 2.

TABLE 2. Test error rates of different parameter initialization methods on ImageNet image classification dataset.

Method type	Top-1Test error rates	Top-5 Test error rates
Xavier model parameter initialization method	25.39±0.23%	13.58±0.16%
The proposed model parameter adaptive initialization method	21.54±0.19%	10.37±0.13%

It can be seen from the experimental results that the model parameter adaptive initialization method proposed in this paper is still effective on large-scale image datasets, which can improve the image classification performance of deep convolutional neural networks, which further verify that the proposed model parameter initialization method has good generalization performance.

IV. ACTION RECOGNITION CLASSIFIER BASED ON SUPPORT VECTOR MACHINE

The proposed method is divided into the following two parts: one part trains a two-level neural network for each action category, and the other builds an SVM classifier. In the training stage of the SVM classifier, for each action category, the training sample of the action is used as a positive sample ($y_i = +1$), and the training samples of the other action categories are used as a negative sample ($y_i = -1$) to train the SVM classification. The parameter vector ρ and slack variable ξ_i are then optimized by minimizing the SVM's objective function [40]. However, when using this idea to train the SVM model, it was found that the SVM's classification result was affected by the imbalance in the number of positive and negative samples in the training set. In the process of training the SVM classifier for each action category to solve the problem of data imbalance, the proposed method uses different penalty parameters C for the positive and negative samples in the training set. Since the number of positive

samples p in the training set is smaller than the number q of negative samples, a larger penalty coefficient C is used for the positive samples. Therefore, the importance of each positive sample data in the training set is increased during the training process. For negative samples, a smaller penalty coefficient C is used. The objective function used to train the SVM classifier is as follows:

$$\min_{\omega,\xi} \frac{1}{2}\|\omega\|^2 + C + \sum_{i=1}^{p}\xi_i + C - \sum_{j=p+1}^{p+q}\xi_j$$
$$s.t.\ y_i\left[\left(\omega^T H_i\right) + b\right] \geq 1 - \xi_i, \quad i = 1, 2, \cdots, p+q$$
$$\xi \geq 0 \tag{36}$$

Where H_{i} is the space-time feature of the $i-$th action sample, $(H_{\mathrm{i}}, y_{\mathrm{i}})$ is the input vector of the SVM classifier, and $p+q$ is the number of training samples used by the training SVM. In the experiment, lib SVM [41] was used to solve the training problem of the SVM classifier. In the experiment, a different penalty coefficient C was set by the ratio of the number of positive and negative samples.

In summary, an SVM classifier F is trained for each action category. Thus, each action category can be represented as an action model $(\theta, W, a, b,)$ consisting of a two-layer neural network and an SVM classifier. This model is used to identify specific action. Of course, multiple action models are used to classify and identify multiple action types.

V. EXPERIMENT ANALYSIS

A. UCF SPORTS ACTION DATABASE

In this experiment, the UCF Sports Human Action Database proposed by the Computer Vision Research Center of the University of Florida was used for analysis and discussion. The database contains 10 categories of action. The relevant action videos are from BBC, ESPN and other sports broadcast channels [44]. The UCF Sports dataset contains 150 videos containing ten actions. The database contains a total of 150 action video sequences. The shooting scenes of these videos are different for different action categories, and the shooting perspective is very wide. These ten actions are Diving-side, Walk, Kicking, Lifting, Riding-Horse, Swing-SideAngle, Skate-Boarding, Swing-Bench, Golfing and Running. The video in the UCF Sports database is shown in Fig. 4.

It is very challenging to identify the motion action in these chaotic real scenes collected from various perspectives. This experiment used the same experimental setup as in [43] and [44] to divide the UCF Sports Action database into a training set of 103 video samples and a test set of 47 action samples. According to [45], this segmentation method can reduce the background correlation between the training set and the test set. Due to the small number of training samples, the proposed method uses a data expansion method to increase the number of video samples in the training set.

In this database, the following three sets of experiments were designed:

FIGURE 4. UCF Sports database part of the action of the image.

1) RECOGNITION EXPERIMENTS FOR SPECIFIC ACTION CATEGORIES

In this experiment, each action tracking sequence was divided into 2×6 video blocks. The effect of the parameter K on the experimental results was evaluated through experiments. The parameter K was set to 300. K is the number of output neurons of each RBM in the first layer network of the neural network architecture. Our proposed algorithm is compared with the reference [43], [44] action recognition algorithm to the recognition rate in each action category. The recognition results are shown in Fig. 5 and Table 3. For SVM, the penalty coefficient $C = 10$, and other parameters such as slack variables are obtained by adaptive matching with data. The neural network parameters are obtained by adaptive matching with the image data to be processed.

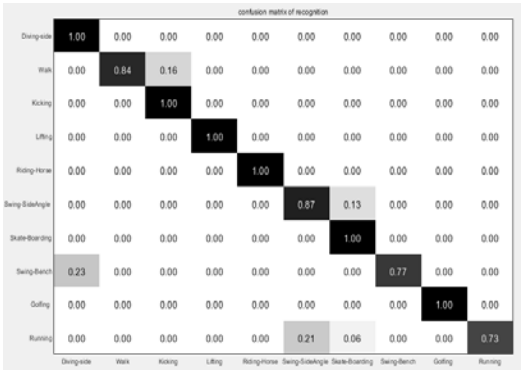


FIGURE 5. UCF Sports Action dataset classification results.

As shown in Fig. 5, the method proposed in this paper correctly identified the rotation action of the UCF Sports Action dataset such as Diving-side, Lifting, Kicking, Riding-Horse, Swing-SideAngle and Skate-Boarding. The correct recognition rates for Golf, Running, Walk and Swing-Bench were 84%, 87%, 77%, and 73%, respectively. The average

TABLE 3. Comparison of the action average recognition rate of the UCF Sports Action database.

Method type	Average recognition rate
BOW[45]	67.4%
FCM[46]	73.1%
DAP[45]	79.4%
Method of this paper	92.1%

recognition rate of the ten activities in the UCF Sports Action dataset was 92.1%.

2) K PARAMETER EXPERIMENT

In the second experiment conducted on the database, the effect of the number of output neurons of the RBM in the first layer of the neural network on the action recognition experiments was observed by adjusting the value of the parameter K. In this paper, the number of output neurons in the second layer of the neural network is $S = 1/3 \times s_w \times s_h \times K$. In Fig. 6, the number K of the output neurons of each RBM of the first layer of the neural network is shown, which influenced the average recognition rate of the action in the UCF Sports Action database. As shown in the Fig. 6, the value of K had a direct impact on the result of the action recognition. This is because the value of K directly determines the number of space-time feature bases of neural network learning designed in this paper. In an experiment to identify motion action in the UCF Sports Action database, the parameter $K = 300$.

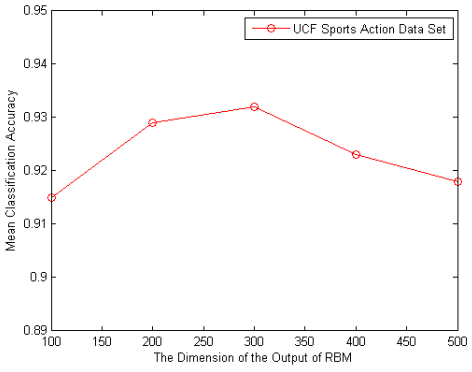


FIGURE 6. Effect of the number of output neurons of RBM on the average recognition rate of the first layer of the neural network architecture.

3) MULTI-TYPE ACTION RECOGNITION EXPERIMENT

The proposed method uses a one-against-all SVM classification strategy to implement the recognition experiment for multiple types of action to compare with the classic action recognition algorithm. According to the neural network and SVM classifier model for each type of action training, after calculating the video block shape features of the action video in the test set, they were input into the neural network and SVM classifier corresponding to each action category. Then, after comparing the classification values of the respective classifiers, an action class corresponding to the classification model having the largest classification value was output, which served as an action label of the test video in the multi-class action recognition experiment.

In this experiment, the generated act tracking sequence was detected from the action video of the UCF Sports Action database and divided into 2×6 video blocks. In addition, the parameter K was set to 300. The comparison between the experimental results of the multi-type action recognition and the recognition effects of other multi-type action recognition algorithms is shown in Table 4. For SVM, the penalty coefficient $C = 10$, and other parameters such as slack variables are obtained by adaptive matching with data. The neural network parameters are obtained by adaptive matching with the image data to be processed.

As shown in Table 4, the average recognition rate of the algorithm proposed in the UCF Sports Action dataset improved by approximately 17.2% compared with the Mironica algorithm [48]. Mironică *et al.* combined the color histograms and HoG features of each frame of the video and used random forests to cluster these data to find cluster centers. Then, it encoded all the frames in the video with a modified local aggregation vector method. The encoded vector was then used as the action recognition feature vector for the video. This method uses a large number of local features in the video and then reduces the computational complexity by encoding the data. However, the essence of this method is still the hand-made local features, and for the cleverly manufactured and stitched local features, a simple encoding operation will result in the loss of effective information. The proposed algorithm operates on four image sequences with different focus points, uses deep learning to learn features, and then performs SVM classification. Therefore, the obtained action recognition features are more targeted than those of Mironica *et al*. At the same time, the method's action recognition rate is also higher.

TABLE 4. Comparison results of multiple types of action recognition experiment results in the UCF Sports Action database.

Method type	Average recognition rate
Reference [48]	74.1
Reference [49]	85.1
Reference [50]	86.5
Reference [47]	88.6
Method of this paper	91.3

As shown in Table 4, the average recognition rate of the proposed algorithm on the UCF Sports Action dataset is 6.2%, which is higher than that of Souly and Shah's algorithm. Souly and Shah used video image intensity values to learn and reasoning to calculate the corresponding corner maps of the video and then clipped and improved the features of the video such as HoG and pouches through the corner maps. Finally, the processed features were used as action recognition features in the video [49]. The essence of this method is to improve the hand-made features in the video by

learning to remove the redundant information and strengthen the effective information. However, hand-made features can only cover the information considered, which is a limitation. Because the features obtained by the feature learning using deep learning in this paper contained richer action information, the accuracy of the action recognition obtained by using this method was better than the algorithm proposed in [49].

As shown in Table 4, the average recognition rate of the algorithm proposed in the UCF Sports Action dataset was approximately 4.8%, which is higher than that of Le *et al.* [13]. Le *et al.* constructed an independent subspace analysis model of a two-layer neuron structure for the boundary features of static images through machine learning. The parameters in the model were obtained through the mass projection gradient descent method. Then, using the stack convolution technique to apply this model to the video image sequence, a human action recognition feature that is sensitive to the boundary speed and rotation angle sensitivity and the position or translation of the boundary was obtained. This human action recognition feature was obtained through learning. It is more versatile than hand-made features and has better extraction features. However, the ontology of the method learning is the boundary features in the image, and the feature types are relatively single. In this paper, the deep learning theory was used to extract the spatiotemporal features in the process of action, and the feature types were more abundant. Therefore, the method proposed in this paper had better performance and higher accuracy than the method of Le *et al*.

As shown in Table 4, the average recognition rate of the proposed algorithm on the UCF Sports Action dataset is improved by approximately 2.7% compared with the algorithm of Rezazadegan *et al.* [47]. Rezazadegan *et al.* calculated a optical flow map corresponding to each image in the video and used the EdgeBoxes algorithm [50] in the optical flow map to obtain a plurality of possible candidate foreground regions in the image. Then, the sum of the amplitudes of the pixel values of the optical flow map in each area was calculated, and the area where the optical flow moved the most was selected as the foreground area of the image. Finally, the image blocks in the corresponding regions in the color image and the optical flow image were sent to the trained convolutional neural network, and then the temporal features were integrated with the spatial features; finally, the human action recognition features corresponding to the video were obtained. The method uses the amplitude of the optical flow map to select the foreground region from multiple candidate regions, resulting in poor robustness. For example, in the act of ''Kicking'', the key position is in the leg. However, when the human body is kicking, the upper body, especially the upper arm, also swings considerably. The proportion of foreground pixels in the rectangular area including the upper body is larger than that in the lower body. Therefore, the algorithm calculates that the amplitude of the optical flow of the pixel in the upper body region of the human body as relatively large, and thus mistakenly selects the upper body region as the foreground region. The proposed algorithm uses deep learning to learn gesture information in the process of action. It can ensure that the region contains a more complete body part, including the key areas of the exercise action. It is more targeted to the target area, so as not to lose the key information of the human action. Therefore, the human action recognition results obtained using the proposed method are superior to those of Rezazadegan *et al*.

FIGURE 7. Sample of KTH action database.

B. KTH ACTION DATABASE

To further verify the action recognition effect of the proposed algorithm; this experiment conducted an action recognition experiment in the KTH action database. The KTH action database [51] contains six action categories, namely, walking, jogging, running, boxing, hand-waving and hand-clapping. These six actions were performed by 25 different people and captured in four different scenarios. The four shooting scenes were an outdoor scene ''outdoors'' (d1), an outdoor scale changing scene ''outdoors with scale variation'' (d2), an outdoor costume changing scene ''outdoors with different clothes'' (d3) and an indoor scene ''indoors''. (d4). The database contains a total of 2,391 action video sequences with a video resolution of 160×120. An example of a sample KTH action database action is shown in Fig. 7. Fig. 8 shows the confusion matrix resulting from the action classification recognition experiments for all scenes in the KTH action database. In the experiment, the size parameter of each cell in the displacement space was set to $N = 5$. The specific recognition results are shown in Table 5. To illustrate the effectiveness of the method in the recognition of multiple types of action, the four scenes of the database were used as a large dataset to conduct action recognition experiments. For SVM, the penalty coefficient $C = 10$, and other parameters such as slack variables are obtained by adaptive matching with data. The neural network parameters are obtained by adaptive matching with the image data to be processed.

In the four scenarios as shown in Fig. 9, the recognition rate of the action ''boxing'' was relatively low. Thus, it is similar to hand-waving and hand-clapping. According to Table 5, the proposed algorithm is better than other action recognition algorithms based on single-class action classifiers in some scenarios, and other scenes are also relatively indistinguishable from those of single-class action classifiers. Moreover, the proposed algorithm is a true multi-type action

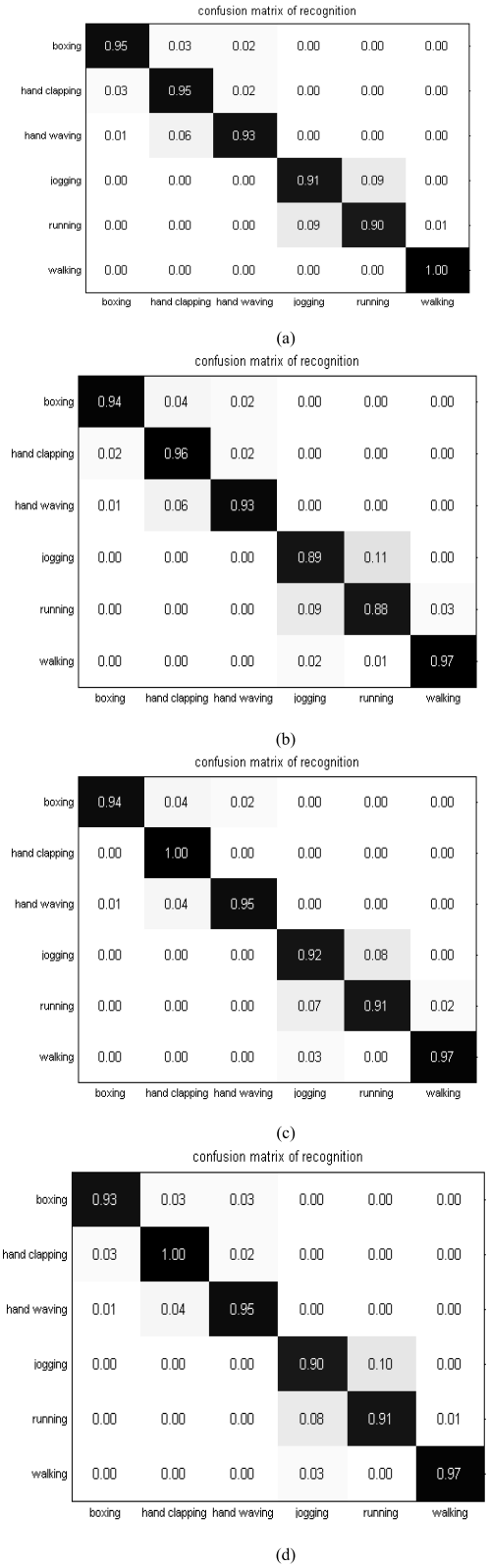


FIGURE 8. The confusion matrix of the KTH action database classification. (a) d1 scene. (b) d2 scene. (c) d3 scene. (d) d4 scene.

recognition algorithm. In the performance of the action recognition, the algorithm is not higher than those based on multiple single-category action classifiers.

TABLE 5. Comparison of action recognition algorithms in the four scenarios of the KTH database.

Method type	d1	d2	d3	d4
Reference [53]	96.8	85.2	92.3	85.8
Reference [54]	93.0	81.1	92.1	96.7
Reference [55]	94.4	84.8	89.8	85.7
Method of this paper	94	92.8	94.8	94.3

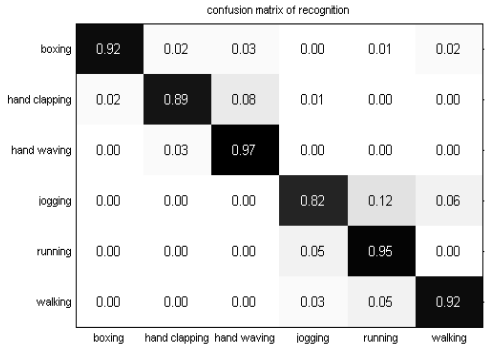


FIGURE 9. The confusion matrix for action recognition in the KTH action database.

TABLE 6. Comparison of recognition performance of the recognition algorithm on the KTH database.

Method type	Average recognition rate
Reference [55]	88.8%
Reference [56]	83.3%
Reference [57]	80.9%
Method of this paper	91.2%

Experiments are conducted to identify all the data in the four scenes of the KTH action database as a whole. The experiment selected the 19 performers' video data as the verification set and the 5 action performers' video data as the training set. Other action videos served as test sets. The confusion matrix for this action recognition experiment is shown in Fig. 9. The recognition effect of the proposed algorithm and other action recognition algorithms on the KTH action database is shown in Table 6. In Fig. 9 and Table 6, it can be seen that the proposed algorithm has better performance in action recognition.

C. SUB-JHMDB ACTION DATABASE

The sub-JHMDB [56] contains 316 video segments containing twelve types of action. These twelve actions are catching, climbing stairs, playing golf, jumping, kicking, picking up, pull-ups, pushing, running, pitching, playing baseball, and walking. In the dataset, there are three training/testing set segmentation methods. This section used the third one to perform the experiments. Among these, 224 video segments were used for training, and the remaining 92 video segments were used for testing. Fig. 10 shows the images in some videos in the sub-JHMDB. The specific action recognition effects are shown in Fig. 11 and Table 7. For SVM, the penalty coefficient $C = 10$, and other parameters such as slack

FIGURE 10. Images from some videos in the sub-JHMDB.

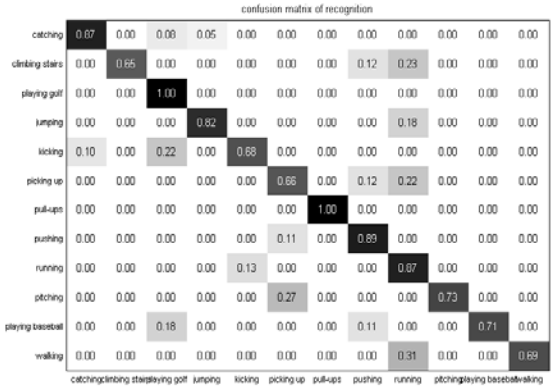


FIGURE 11. sub-JHMDB classification results.

TABLE 7. Comparative experimental results of the sub-JHMDB.

Method type	Average recognition rate
Reference [59]	61.9%
Reference [60]	62.5%
Reference [62]	69.3%
Method of this paper	79.8%

variables are obtained by adaptive matching with data. The neural network parameters are obtained by adaptive matching with the image data to be processed.

As shown in Fig. 11, the proposed method in this paper can accurately identify playing golf, pull-ups, pushing, running, catching, jumping, pitching, and playing baseball in the sub-JHMDB. The corresponding action recognition rates were 100%, 100%, 89%, 87%, 87%, 82%, 73%, and 71%; recognition rates for walking, kicking, picking up, and climbing stairs were 69%, 68%, 66%, and 65%. Because the action in the sub-JHMDB is more complex than the action in the UCF Sports Action dataset, the accuracy of this action recognition is low.

As shown in Table 7, the comparative statistics of the recognition rate of the action of this method and other mainstream algorithms on the sub-JHMDB. As shown in Table 7, the average recognition rate of our proposed algorithm in the sub-JHMDB was better than that of Richard *et al*. The average action recognition rate of the method proposed by [57] increased by 17.9%. Richard *et al*. improved the classification word package method to train a recursive neural network that can recognize and supervise the generation of visual vocabulary and used a support vector machine to train the classifier to achieve the purpose of identifying and classifying human action. In this method, the input data of the neural network is a complete original image frame, without considering that the different parts of the video image contribute differently to the action recognition, and the method of separately processing the image does not consider the time information in the video. The proposed method can use the deep learning theory of initializing model parameters to learn video images actively and can obtain more abundant action recognition information.

The average recognition rate of the proposed algorithm on the sub-JHMDB is 17.3% higher than the average action recognition rate proposed by Gkioxari and Malik [58]. Malik used a method of selecting and searching for objects in color images [59] to generate two thousand candidate image block regions for each image of the video and then removed the regions in the region where the pixel amplitude of the corresponding optical flow graph was smaller than the threshold value. The corresponding color image blocks and optical flow image blocks were cut out with the remaining regions and sent to a convolutional neural network for processing, and the vectors obtained by all video image blocks were connected to obtain the action recognition characteristics of the video. The method selects the key area in the image by calculating the optical flow value in the image block area. Its problems are similar to those of the Rezazadegan *et al*.'s approach, which can easily lead to the miss election of key areas. The proposed method uses deep learning to learn pose information in the process of action, which can ensure that the region contains a more complete body part, ensuring that it contains the key areas of the exercise action, and avoids the loss of key information of the human action. Therefore, the accuracy of the human action recognition results it obtains was higher than the method proposed by Malik.

As shown in Table 7, the average action recognition rate of the proposed algorithm on the sub-JHMDB was 10.5%, which is higher than the average action recognition rate proposed by Peng *et al.* [60]. Peng *et al.* used a multi-level nested Fisher vector coding method to obtain action expression features in video. In this method, the improved dense trajectory feature of the input video is obtained by sampling

in the first layer, and the original Hsher vector is obtained by compressing it with Fisher coding at the video segment level. Since the original Hsher vector feature dimension obtained in the first layer was too high, the data were projected using the maximum edge learning method in the second layer, and then the processed data was further Fisher encoded. The vector obtained after the second encoding was used as a feature vector for human action recognition. The method improves the coding method of the video-intensive trajectory features and uses the cascaded secondary encoding method to extract the features hierarchically, which is more refined than the features obtained by using only one coding. However, this method is still based on the hand-made features in nature. Its self-learning ability is far lower than that of the deep learning theory used in this paper. Therefore, the human speech recognition accuracy rate obtained by this method is better than the methods proposed by Peng *et al*.

D. UCF101 ACTION DATABASE

The UCF101 dataset [61] is a collection of human behavior data provided by the Center for Computer Vision Research at the University of Florida. The dataset is an extension of the UCF50 dataset. It contains 101 human behaviors. Each behavior category is divided into 25 groups. Each group contains 4-7 videos of the same type, with a total of 13320 segments of behavioral video. The dataset video was divided into three sets of training/test videos [61] during the behavioral recognition experiment: split1, split2, and split3. In this experiment, relevant experiments were carried out according to the standard, and the average action recognition rate of the three sets of test videos was taken as the final experimental result. For SVM, the penalty coefficient $C = 10$, and other parameters such as slack variables are obtained by adaptive matching with data. The neural network parameters are obtained by adaptive matching with the image data to be processed.

In order to better reflect the effectiveness and advantages of the proposed algorithm, the UCF101 behavior database is identified by the method, the MF algorithm in [62], and the EMV-CNN algorithm in [63]. The algorithm running speed test includes video decoding to obtain motion features and motion vectors compensated by performing the algorithm. They are implemented on Intel Core, and the [62], [63] convolution feature extraction part is obtained on CPU-based Caffe. The test running speed results of 5,000 samples randomly sampled on each data set are averaged. The speed test includes complete action recognition process such as video dump decoding, feature extraction and behavior classification. The specific action recognition effect is shown in Table 8.

As shown in Table 8, In the UCF101 dataset, the proposed algorithmic recognition rate of this paper is 6.1%, 1.9%, and 1.3% higher than that of the [62], [63], and the neural network for action recognition classification. Although the algorithm runs faster than the reference [63], the gap is small and can meet the real-time processing application requirements.

TABLE 8. Comparative experimental results of the UCF101.

Method type	Average recognition rate	Running speed
Reference [64]	82.2%	698.4 fps
Reference [65]	86.4%	535.2 fps
Method of this paper + neural network behavior classification	87.0%	539.9 fps
Method of this paper	88.3%	536.3 fps

E. HMDB51 ACTION DATABASE

The HMDB51 data set was provided by Kuehne *et al*., Brown University, USA. The data set contains a total of 51 behavior categories, each of which contains at least 100 videos, including 6676 action sequences. The data set video is also divided into 3 training/test splits when performing action recognition experiments. Each type of behavioral video for each group contains 70 training videos and 30 test videos. This paper is set according to the reference [64], based on the average recognition rate of the three groups as the final result.

In this section, the method, the neural network for action recognition classification, the GME algorithm in reference [65] and the EMV-CNN algorithm in reference [63] are used to identify the behavior of the HMDB51 behavior database. The algorithm runs at an Intel Core i5-5200 (2.2 GHz) CPU. The results of random sampling of 5000 samples on each data set are averaged, and the speed test objects are the running speeds of the three algorithms. The specific action recognition effect is shown in Table 9.

TABLE 9. Comparative experimental results of the HMDB51.

Method type	Average recognition rate	Running speed
Reference [67]	46.7%	652.2 fps
Reference [65]	50.6%	576.5 fps
Method of this paper + neural network behavior classification	53.7%	581.7fps
Method of this paper	55.2%	578.4 fps

As shown in Table 7, the action recognition rate of the algorithm in the HMDB51 data set is 8.5%, 4.6%, and 1.5%, which higher than that of the [65], [63], and neural network for action recognition. Although the speed of the algorithm in this paper is lower than that in [63], the difference is small. Compared with the [63], the use of neural networks for action recognition has improved, but they are all in an operational level and can meet the needs of real-time processing applications.

VI. CONCLUSION

In this paper, a neural network model parameter adaptive initialization method deep learning model based on the MMN activation function and the Maxout activation function is proposed. Based on this, a method for global temporal and

spatial feature learning of human action based on video angle is proposed. The deep learning model proposed in this paper cannot only guarantee the stable propagation of gradients in different hidden layers but also accelerate the convergence speed of neural network model training and avoid the slow convergence of training caused by inappropriate initialization of the model. In addition, it can improve the image recognition performance of the deep convolutional neural network model.

The human action recognition method mentioned in this paper does not directly extract features from the pixel information of the video but learns more abstract and higher-level space-time features from the underlying shape feature sequences that characterize the action motion laws. The basic idea is to design a two-layer neural network structure using training and a support vector machine to characterize a specific action model. The method utilizes a neural network based on a restricted Boltzmann machine to achieve self-learning of the discrete distribution of action movement laws or action movement information, thereby realizing the recognition of certain types of human action. Although the algorithm is designed to identify certain types of action, the algorithm can still be used for multi-class action recognition with a one-to-many SVM classification strategy.

This paper used three human action datasets to verify the proposed algorithm and compared it with several existing mainstream algorithms. The experimental results showed that the proposed method can not only quickly and accurately identify a certain type of human action but can also identify multiple types of human action. Its recognition effect is better than other existing types of human-specific action recognition algorithms and human body multi-type action recognition algorithms.

REFERENCES

[1] M. T. Harandi, C. Sanderson, S. Shirazi, and B. C. Lovell, "Kernel analysis on Grassmann manifolds for action recognition," *Pattern Recognit. Lett.*, vol. 34, no. 15, pp. 1906–1915, 2013.

[2] H. Rahmani, A. Mian, and M. Shah, "Learning a deep model for human action recognition from novel viewpoints," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 667–681, Mar. 2018.

[3] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1510–1517, Jun. 2018.

[4] X. Peng, L. Wang, X. Wang, and Y. Qiao, "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice," *Comput. Vis. Image Understand.*, vol. 150, pp. 109–125, Sep. 2016.

[5] J. C. Niebles, H. Wang, and L. Fei-Fei, "Unsupervised learning of human action categories using spatial-temporal words," *Int. J. Comput. Vis.*, vol. 79, no. 3, pp. 299–318, 2008.

[6] C. Thurau and V. Hlavac, "Pose primitive based human action recognition in videos or still images," in *Proc. IEEE CVPR*, Jun. 2008, pp. 1–8.

[7] T. Qi, Y. Xu, Y. Quan, Y. Wang, and H. Ling, "Image-based action recognition using hint-enhanced deep neural networks," *Neurocomputing*, vol. 267, pp. 475–488, Dec. 2017.

[8] B. B. Amor, J. Su, and A. Srivastava, "Action recognition using rate-invariant analysis of skeletal shape trajectories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 1–13, Jan. 2016.

[9] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proc. IEEE CVPR*, Jun. 2014, pp. 28–37.

[10] D. Wu and L. Shao, "Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition," in *Proc. IEEE CVPR*, Jun. 2014, pp. 724–731.

[11] Z. Liang, X. Wang, R. Huang, and L. Lin, "An expressive deep model for human action parsing from a single image," in *Proc. Int. Conf. Multimedia Expo*, Jul. 2014, pp. 1–6.

[12] M. Blank, L. Gorelick, E, Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2005, pp. 1395–1402.

[13] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *Proc. IEEE CVPR*, Jun. 2011, pp. 3361–3368.

[14] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 140–153.

[15] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 495–502.

[16] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, 1951.

[17] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 161–168.

[18] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1026–1034.

[20] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. ICML*, 2013, pp. 1302–1319.

[21] Z. Jiang, Z. Lin, and L. S. Davis, "Recognizing human actions by learning and matching shape-motion prototype trees," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 533–547, Mar. 2012.

[22] S. Zhang, H. Yao, X. Sun, and X. Lu, "Sparse coding based visual tracking: Review and experimental comparison," *Pattern Recognit.*, vol. 46, no. 7, pp. 1772–1788, Jul. 2013.

[23] S. Zhang, H. Yao, H. Zhou, X. Sun, and S. Liu, "Robust visual tracking based on online learning sparse representation," *Neurocomputing*, vol. 100, pp. 31–40, Jan. 2013.

[24] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.

[25] D. D. Dawn and S. H. Shaikh, "A comprehensive survey of human action recognition with spatio-temporal interest point (STIP) detector," *Vis. Comput.*, vol. 32, no. 3, pp. 289–306, Mar. 2016.

[26] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[27] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," in *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*, vol. 1. 1986, pp. 194–281.

[28] Y. Freund and D. Haussler, "Unsupervised learning of distributions on binary vectors using two layer networks," Univ. California, Santa Cruz, CA, USA, Tech. Rep., 1994.

[29] M. Á. Carreira-Perpiñán and G. E. Hinton, "On contrastive divergence learning," in *Proc. CCVP*, 2005, pp. 97–106.

[30] D. Mishkin and J. Matas. (2015). "All you need is a good init." [Online]. Available: https://arxiv.org/abs/1511.06422

[31] D. Hendrycks and K. Gimpel, "Generalizing and improving weight initialization," pp. 1607–1618, 2016.

[32] S. Ma, S. A. Bargal, J. Zhang, L. Sigal, and S. Sclaroff, "Do less and achieve more: Training CNNs for action recognition utilizing action images from the Web," *Pattern Recognit.*, vol. 68, pp. 334–345, Aug. 2017.

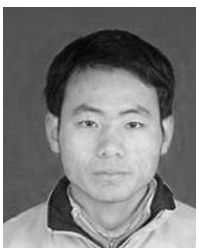
[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[34] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[35] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[36] G. E. Hinton and R. R. Salakhutdinov, "A better way to pretrain deep Boltzmann machines," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 2447–2455.

[37] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer, 2012, pp. 599–619.
[38] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010.
[39] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[41] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. ICML*, 2013, pp. 1302–1319.
[42] J. J. Grefenstette, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*. London, U.K.: Psychology Press, 2013, pp. 378–390.
[43] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27–33, 2011.
[44] M. D. Rodriguez, J. Ahmed, and M. Shah, "Action MACH a spatio-temporal maximum average correlation height filter for action recognition," in *Proc. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 15–23.
[45] M. Raptis, I. Kokkinos, and S. Soatto, "Discovering discriminative action parts from mid-level video representations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 1242–1249.
[46] T. Lan, Y. Wang, and G. Mori, "Discriminative figure-centric models for joint action localization and recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2003–2010.
[47] F. Rezazadegan, S. Shirazi, N. Sünderhauf, M. Milford, and B. Upcroft, "Enhancing human action recognition with region proposals," *Ultrasound Obstetrics Gynecol.*, vol. 32, no. 3, pp. 335–340, 2015.
[48] I. Mironică, I. C. Duţă, B. Ionescu, and N. Sebe, "A modified vector of locally aggregated descriptors approach for fast video classification," *Multimedia Tools Appl.*, vol. 75, no. 15, pp. 9045–9072, 2016.
[49] N. Souly and M. Shah, "Visual saliency detection using group lasso regularization in videos of natural scenes," *Int. J. Comput. Vis.*, vol. 117, no. 1, pp. 93–110, 2016.
[50] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 391–405.
[51] C. Schüldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach," in *Proc. IEEE Int. Conf. Pattern Recognit.*, Aug. 2004, pp. 32–36.
[52] K. Schindler and L. van Gool, "Action snippets: How many frames does human action recognition require?" in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 22–27.
[53] M. Ahmad and S.-W. Lee, "Human action recognition using shape and clg-motion flow from multi-view image sequences," *Pattern Recognit.*, vol. 41, no. 7, pp. 2237–2252, 2008.
[54] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[55] Y. Ke, R. Sukthankar, and M. Hebert, "Spatio-temporal shape and flow correlation for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.
[56] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black, "Towards understanding action recognition," in *Proc. IEEE Comput. Vis. (ICCV)*, Dec. 2013, pp. 3192–3199.
[57] A. Richard and J. Gall, "A bag-of-words equivalent recurrent neural network for action recognition," *Comput. Vis. Image Understand.*, vol. 156, pp. 79–91, Mar. 2017.
[58] G. Gkioxari and J. Malik, "Finding action tubes," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 759–768.
[59] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, Apr. 2013.
[60] X. Peng, C. Zou, Y. Qiao, and Q. Peng, "Action recognition with stacked Fisher vectors," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 581–595.
[61] K. Soomro, A. R. Zamir, and M. Shah. (2012). "UCF101: A dataset of 101 human actions classes from videos in the wild." [Online]. Available: https://arxiv.org/abs/1212.0402
[62] V. Kantorov and I. Laptev, "Efficient feature extraction, encoding and classification for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 2593–2600.
[63] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang, "Real-time action recognition with enhanced motion vector CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2718–2726.
[64] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2556–2563.
[65] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 3551–3558.



FENG-PING AN received the B.S. degree from the School of Economic and Management, Hefei University, Hefei, China, in 2008, the M.S. degree from the School of Economics and Management, Hebei University of Engineering, Handan, China, in 2011, and the Ph.D. degree from the School of Computer and Communication Engineering, Beijing University of Science and Technology. He was with Huaiyin Normal University. He was a Post-Doctoral Researcher with the Beijing Institute of Technology. His research interests include image processing, artificial intelligence, and pattern recognition.