Dynamic Hand Gesture Recognition Using Multi-direction 3D Convolutional Neural Networks

Jie Li, Mingqiang Yang, Yupeng Liu, Yanyan Wang, Qinghe Zheng and Deqiang Wang

Abstract—**The static hand gesture recognition under simple or complex background has become mature, but the dynamic hand gesture recognition against simple background is still challenging. The reason is that we need pay attention to not only spatial features but also temporal features in dynamic hand gesture recognition. In this paper, we mainly research the drivers' dynamic hand gesture recognition applied in automotive users interfaces to ensure the safe and improve the comfort. We use 3D convolutional neural networks(3DCNN) to classify the dynamic hand gesture, which can effectively extract temporal-spatial features. Our contributions have three aspects: 1)We extract the key frames from the origin video by computing the optical flow of two adjacent frames; 2)We propose a new data augmentation method–data temporal cropping, which can improve the recognition accuracy and prevent overfitting effectively. 3)We propose a novel multi-direction convolutional neural networks(mdCNN), which can extract the distinguished temporal-spatial features from the short videos. In the end, we present experiment results based on VIVA dataset and achieve recognition accuracy of 65.35%, which is higher than other methods.**

Index Terms—**Hand gesture recognition, Temporal-Spatial features, Automotive users interfaces, 3D convolutional neural networks**

I. Introduction

Dynamic hand gesture recognition is a challenging research hotspot in the filed of computer vision, which is widely used in various human-computer interface applications. The hand gesture also has extensive applications in our life. For example, deaf and dumb people communicate mainly by gestures, and people in different languages also communicate with each other by gestures. The traffic police use different gestures to command traffic. So the gesture language is called second language for human. Compared with face and other biosignatures, the advantage of gesture is more lively, nature and convenient. In recent years, hand gesture recognition as a human-computer interaction(HCI) application has been widely used in many fields such as smart television, virtual reality, intelligent robot, etc. In this paper, we build a dynamic gesture recognition system for touchless interfaces in cars. The touchless interface can improve safety and comfort for driver because it allows the driver to pay more attention to the driving while interacting with other devices by using gestures, such as vehicle navigation,audio, etc. The gesture recognition algorithm can be divided to static gesture recognition and dynamic gesture recognition in car. For static gesture, we just care about the spatial features of hand, Zhuang et al. [1] used combinational features and compressive sensing to classify the static hand gesture. In the dynamic gesture algorithm, we extract not only spatial features but also temporal features. Therefore, The hand pose in spatial and hand trajectories in temporal are both worth concern. The dynamic hand gesture recognition is a more challenging research direction.

In the last decade, many researchers have been working on the research of dynamic gesture recognition and have made great achievements [2]–[5]. Before the deep learning eve, researchers mainly worked on designing better feature descriptors to improve the recognition accuracy. So a lot of hand-craft spatiotemporal [6] feature extraction methods were introduced. Histogram of Oriented Gradient(HOG) was proposed by Navneet Dalal and Bill Triggs in [7], which was firstly used to detect the pedestrian in single frame and then extended to other image processing field. For dynamic gesture recognition, HOG chooses a vectorization operator on the spatial descriptors by statistic the gradient information on each frame. HOG2 is another feature descriptor which applies the HOG twice(one in spatial domain and another in temporal domain). Klaser et al. [8] proposed histogram of 3D oriented gradients(HOG3D) method to extract spatiotemporal features. Ohn-Bar and Trivedi et al. [9] evaluated these hand-craft spatiotemporal features on VIVA dataset which includes RGB data and Depth data. The best recognition accuracy is 64.5% with a combination of HOG and HOG2 features by using an SVM classifier [10]. In [11], Nowozin and Shotton put forward the concept of action points and used Hidden Markov Model(HMM) [12] as natural temporal anchors of human actions. Wang et al. [13] proposed a elaborated discriminative hidden-state method to recognize hand gesture. In addition, Heng et al. [14] introduced dense trajectories and motion boundary descriptor which used the optical flow to extract dense trajectories. As the best temporal-spatial feature descriptor, this algorithm got the best result in human action recognition in traditional hand-craft feature methods. However, dynamic hand gesture recognition has been facing huge challenge with the various illumination and different subjects [15]–[17].

In this paper, our contributions consist of three parts:1.we design a method based on optical flow to extract the key frames from the origin video, which can save the neural network training time; 2. we propose a data augmentation method for video in temporal domain to against the network overfitting; 3. the most important part is that we build a multi-direction convolution neural network(mdCNN) architecture which improves temporal-spatial feature extraction ability on the basis of the origin 3D convolution neural network. As

Manuscipt received January 26, 2019; revised March 25, 2019. This work was supported by National Key R&D Program of China under grant 2018YFC0831503, National Nature Science Foundation of China(61571275)and Fundamental Research Funds of Shandong University(31410078611043)

Jie Li is with the School of Information of Science and Engineering, Shandong University, Qingdao 266200, China (email: stu_jie_li@163.com).

Mingqiang Yang*(as corresponding author), Yupeng Liu, Yanyan Wang, Qinghe Zheng and Deqiang Wang are with the School of Information of Science and Engineering, Shandong University, Qingdao 266200, China (*email: imageinstitute@outlook.com).

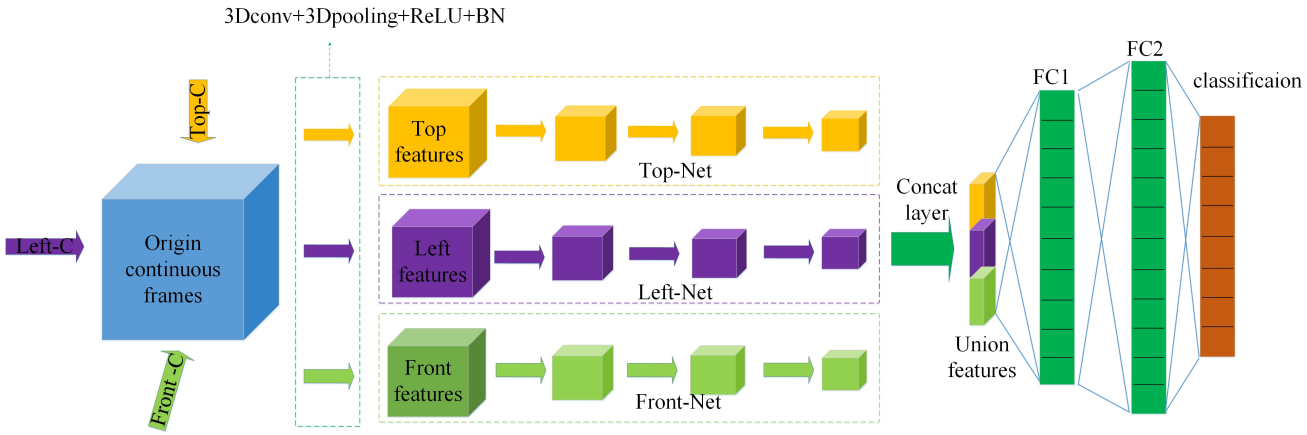


Fig. 1: Architecture of our model. In figure, the inputs of our model is a video volume and our model consists of three sub nets, which are used to extract spatial-temporal features from different directions, and then three sub nets are fused in the next layers.

shown in Fig. 1, some continuous frames can be seen as a cubic. We apply 3D convolutional kernels to this cubic on three directions(front, top and left), and then generated three temporal-spatial features. We adapt two ways to processing these features, the first way is that we concatenated these three features by a concat layer and then connect with the fully connection layer. The second way is that each feature is connected with the fully connection layer directly and then fusion at score layer. The details will be introduced in the latter sections, and our experiment results show the difference of these two ways on influence of recognition accuracy. The rest of this paper is organized as follows. We first introduce some related works of CNN and dynamic hand gesture recognition based on CNN in Section II. The details of our mdCNN architecture are introduced in Section III. And our key frames extraction and temporal data augmentation are introduced in Section IV. Section V shows our experiment results and analysis. Finally, we draw conclusions and discuss the future works in section VI.

II. Related Works

Hand gesture recognition has been paid more and more attention by researchers recently, especially the dynamic hand gesture recognition. Previous works mainly focused on two aspects: 1. Locate the hand accurately in image. Qingrui Zhang et al. [18] segmented and localized the hand by using saliency and skin color 2. Extract discriminative features from origin data. Qinghe Zheng et al. [19] extracted hand features by combination Gaussian Mixture Model and Partial Differential Equation. In this paper, we only focus on the second part and classify the gesture automatically. In this section, the first part introduces the traditional CNN and 3DCNN. In the second part, we introduce some works based on CNN and 3DCNN.

A. CNN and 3DCNN

In recent years, deep learning [20] become the research hotpot, especially CNN technique which transforms the origin data into the high-level features through the combination of multi nonlinear models. Therefore, the image and other multi-dimensional signals are convenient to be processed by CNN. After Alexnet acquired the champion of Imagenet in 2012 [21], the deep convolutional neural networks has been applied to various image tasks successfully, such as image classification [22] [23], object detection [24] [25] and image retrieval [26]. Meantime, some popular CNN architectures represented by VGGNet [27] and GooLeNet [28] are proposed, which made impressive achievements in variety of image tasks. But in video classification tasks, these networks fail to achieve the desired results, because the traditional CNN only extracts spatial features from a single frame without extracting temporal features from the video. For purpose to extract the temporal-spatial features from the video, Tran et al [29] proposed 3D convolutional kernel, which increases the time dimension compared with the traditional 2D convolutional kernel. They trained 3DCNN on a large scale dataset–UCF101 dataset, and did some experiments to prove that the $3\times3\times3$ kernel is the best kernel size for all 3D convolutional layers. Fig. 3 shows the differences between the CNN and 3DCNN. Fortunately, 3D convolutional layer as a common layer is also used to construct popular network architectures such as VGGNet. However, these popular network architectures can not capture discriminatory features from small data sets because these deep networks will overfit when training on small dataset. The dropout [30] technique can effectively solve the overfitting problem through randomly deactivating some neurons in the fully connenction layer. Beside, batch normalization [31] can not only effectively resist overfitting, but also greatly quicken the speed of network convergence. Above two techniques are applied to our network architecture.

B. CNN-based video and dynamic hand gesture recognition

Inspired by the deep convolutional neural networks achieving great success in image domain [21] in the past few years, many researchers have been trying to apply deep neural networks to video classification such as human action recognition. For purpose of extraction spatial-temporal features from video, Simonyan and Zisserman [32] proposed two-stream convolutional neural networks which extract the spatial features and temporal features through the spatial networks and temporal networks respectively, and then two

networks fused in the score layer. As known, flow algorithm can extract the motion information from the continuous frames. Therefore, the input of the temporal networks are flow images computed by the dense optical flow algorithm. On the basis of the two- stream networks, Wang et al [33] proposed a temporal segment network(TSN), they segmented the origin clips to some short segments and then input these segments to two-stream networks [34]. The TSN architecture got great result on UCF101 dataset for action recognition, but it is not a end-to-end classification net and it sacrifices costly computing resource to generate flow image as well. To solve these problem, Du Tran et al [29] introduced an end-to-end network architecture called C3D, which can effectively extract temporal-spatial features with generality, simplicity and compactness. As an independent field of video classification, dynamic hand gesture recognition based CNN have achieved state of art performance. In [35], Pavlo used C3D network to identify dynamic gestures. Compared with craft-feature methods, this method has gained overwhelming superiority. In [36], a multi-scale and multi-modal CNN architecture was proposed, which achieved the start-of-art performance in the CharLearn 2014 gesture recognition challenge. Pigou [37] used traditional CNN to extract spatial features and then model the spatial features temporally by using the Recursive Convolutional Neural Networks(RNN) [38] with Long Short-Term Memory units(LSTM) [39]. Molchanov et al. [40] fused RGB data,depth data and radar sensors of hand gestures and then fed these information into 2D convolutional neural networks. Neverova et al. [41] proposed a pose descriptor combining RGB-D images around hand regions and trained using 2D convolutional neural networks. In our approach, we employ a mdCNN architecture to extract complete spatial-temporal features, and we use flow method to extract the key frames from origin videos.

III. The details of mdCNN

In this section, we introduce the construction of our mdCNN in detail, including the whole framework and the training methods in the training process.

A. Architecture of mdCNN

mdCNN architecture includes three branches(branch-front, branch-top and branch-left) which have the same network structure(see Fig. 2), and each branch extracts temporal and spatial features from one direction of the original data, respectively. The input of our architecture is a clip which has 8 frames, and in each branch, the first hidden layer is 3D-convolution layer(C1), and then add BN-layer(BN1), Relu layer(R1) and 3D-pooling layer(P1). These four different layers form a block and repeat it four times until they are connected to two fully connected layers. Finally, the dynamic gestures are classified using the SoftMax layer. We design two ways to merge these three branches. The first way is that these three branches merged after softmax layer by averaging three classifiers scores. The second method is to concatenate the three spatiotemporal features extracted from each branch, and then connected to two fully connection layers and softmax layer. The second method is based on feature level fusion, which makes it more scientific and

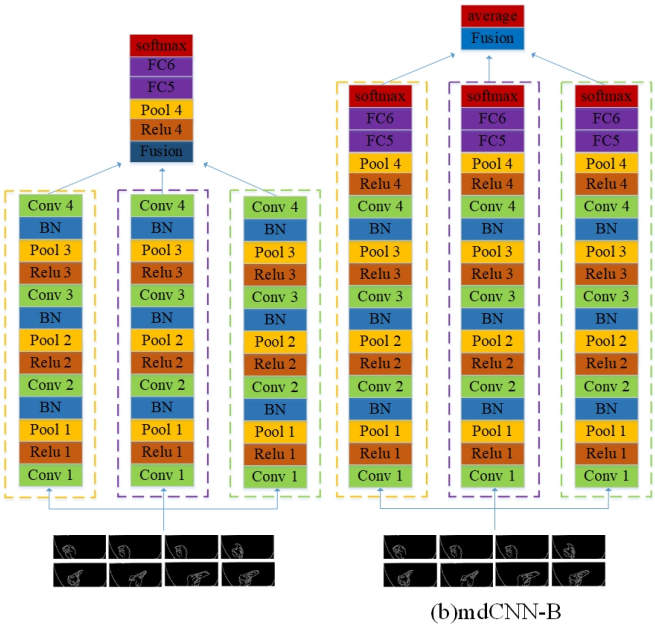
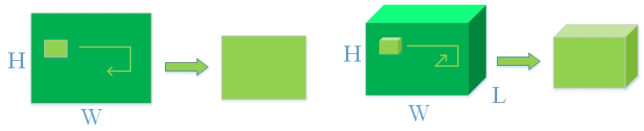


Fig. 2: Our proposed two mdCNN models, mdCNN-A(left) and mdCNN-B(right). The difference of two models is fusion method, in mdCNN-A, three branches are fused previous after Conv4 layer, in mdCNN-B, three branches are fused after softmax layer.

explanatory, and makes up for the shortage of single direction feature extraction, which is also confirmed by experiments. All branches use the same parameter settings except the convolution layers and ReLu layers, and the details of parameter settings are described in detail in the next section

B. 3D convolution layer

We use the video-based 3D convolution layer to replace frame-based 2D convolution, because it can extract extra spatial-temporal features from video. In each 3D convolution layer, there are different numbers of convolution kernels(also called filters). If we set the number of kernel is N, then after convolution operation on the feature map of the previous layer by convolution kernel, N feature maps will be generated in this layer. Due to the convolution operating only perceive local information, features extracted by lower layers are called local features. With the increase of network depth, these local features is synthesised at the higher level and the global features is obtained. Besides, convolution operating with the characteristic of wight sharing, which can effectively reduce the number of parameters to against the overfitting and improve the generality of network. There are two most commonly methods used to initialize convolution kernels. The one is gaussian random initialization performed by gaussian distribution. The another is xavier [42] method related to the number of convolution kernels of the previous layer and current layer. As the number of training iterations increases, these convolution kernels will be changed, and eventually these convolution kernels will be trained into a variety of feature extractors. The 3D convolution operating



(a) 2D convolution (b) 3D convolution

Fig. 3: 2D convolution vs 3D convolution. (a)The result of applying 2D convolution on an image is also an image. (b) Applying 3D convolution on an video volume,the result is also an volume.

is given by

$$v_{ij}^{xyz} = Relu(\sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)} + b_{ij}) \tag{1}$$

where v_{ij}^{xyz} is the value at position (x, y, z) on the jth feature map in the ith layer, P and Q and R are the height,width and temporal size of 3D convolution kernel, w_{ijm}^{pqr} is the (p, q, r)th value of the kernel connected to the mth feature map in the previous layer, m means that $(i-1)^{th}$ layer has m feature maps connected to current layer, b_{ij} is the bias of the j^{th} output feature map in the i^{th} layer. $Relu(\bullet)$ is the activation function, which is given by

$$Relu(x) = max(0, x) \tag{2}$$

C. 3D pooling layer

Similar with the 3D convolution, 3D pooling also extends the 2D pooling on the time dimension and it makes possible to build deeper networks. The main function of pooling layers is to merge local features and reduce the size of feature maps, which makes the filters to get global and contextual information as the network deeper. Max pooling and Mean pooling are two commonly methods in this layer. The former is better than the latter usually, because Max pooling is more like making feature selection, selecting features with better classification recognition and providing nonlinearity. In CNN, the error of feature extraction mainly comes from two aspects. One is that the estimated variance increases due to the neighborhood size constraint; another one is that the error of the convolution layer parameter leads to the deviation of the estimation error. Mean-pooling can reduce the first error, more retain the background information of the image, Max-pooling can reduce the second error, more retain the texture information. In our classification goal, in order to extract the better hand features, we choose the Max-pooling.

D. Batch normalization layer(BN layer)

For deep convolutional networks, there is a challenge question called gradient vanishing caused by chain derivation rule in back propagation, which makes the network training difficultly and convergence slowly. This is a interesting research point in deep learning and many papers have studied this problem, and some practical solutions were proposed. In [21], Alex proposed a novel activity function named Rectified Linear units(ReLU) showed in equation 2. Compared with traditional activity functions such as sigmoid and tanh, the gradient value of ReLU function is one when the inputs is greater than zero, so the gradient vanishing problem has been improved in some extent. In [43], Kaiming He improved the structure of deep convolution neural network and proposed residual network(Resnet) based on the short connection, which made the depth of network reach 152 layers. It effectively solves the problem that the network is difficult to train because of the gradient vanishing, and improves the classification accuracy of network. Qinghe Zheng [44] improved the generalization ability of deep CNN by implicit regularization. Although the above methods improve the problem of gradient vanishing to a certain extent, they have not been solved from the perspective of training data. There is a very important assumption in machine learning: Independent and identically distributed(IID). It is assumed that the training data and the test data satisfy the same distribution, which is the basis for the model obtained from the training data to achieve good results in the test data. The function of batch normalization is that makes the input of each layer of neural network keep the same distribution in the process of deep neural network training. The basic idea of batch normalization is quite intuitive. Because the activation input value of the deep neural network before the non-linear transformation gradually shifts or changes with the deepening of the network or during the training process, the trend of changing is that the input value approach the saturation area of the activation function(such as sigmoid function). And each batch data has different distribution(different mean value and variance), which makes it necessary to use smaller learning rates and better initial weights, finally it leads to network convergency slow. It is also difficult to use saturating nonlinearities activation functions (such as sigmoid, both positive and negative sides will be saturated). This leads to the gradient vanishing in shallow convolution layer of convolution neural network, that is the essential reason why the convergence of training deep neural network is slow. The function of the batch normalization layer is to transform the data into Gaussian distribution with mean of μ and variance of σ before features input to the next convolution layer data. Then the input data will apart from the saturating area of activation function approaching the non-saturating area where it has the greater gradient value, so it can avoid the gradient vanishing and speed up the convergence of the deep convolutional networks. Beside, by adding the BN layer, train data and test data have the identity distribution of Gaussian, it reduces the overfitting of the network, and improves the generalization of the network. The concrete batch normalization transform is shown below.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1, \ldots, x_m\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{//mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{//mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{//normalize}$$

$$y_i \leftarrow \gamma\hat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta} \qquad \text{//scale and shift}$$

In training phase, we get the learned parameters γ and β by multi batch training. In test phase, we utilize these two parameters to compute the distribution of test data, the formula is defined by:

$$y = \frac{\gamma}{\sqrt{Var[x] + \varepsilon}} \cdot x + (\beta - \frac{\gamma \cdot E[x]}{\sqrt{Var[x] + \varepsilon}}) \tag{3}$$

where the $Var[x]$ and $E[x]$ are the unbiased estimation of variance and mean of train set computed by:

$$E[x] \leftarrow E_B[\mu_B] \tag{4}$$

$$Var[x] \leftarrow \frac{m}{m-1} E_B[\sigma^2_B] \tag{5}$$

So after the BN transformation, the training data and test data have the identity distribution, which enhances the network generalization effectively.

IV. Dataset And Preprocessing

In our experiments, we use the VIVA dataset to evaluate mdCNN architecture. In the first part, we introduce the VIVA dataset in detail. Then, we introduce two main preprocessing operations for raw data, including key frames extraction based on Horn-Schunck algorithm [45] and data augmentation. Experiments show that these processing achieve better experimental results.

A. VIVA Dataset

The VIVA challenge dataset contains 19 types of dynamic hand gestures with a total 885 depth and intensity video sequences performed by 8 subjects in vehicle with varying illumination conditions, includes Swipe R(shown in Fig. 4), Swipe L, Swipe D, Swipe U, Swipe V, Swipe X, Swipe +, Scroll R, Scroll L, Scroll U, Tap-1, Tap-3, Pinch, Expand, Rotate CCW, Rotate CW, Open, Close. Each subject includes two dynamic hand gesture performance involved hand or finger motion, one in the drivers seat using right hand, the other in the front passengers seat using left hand. All these videos has a resolution of 115×250 captured by Microsoft Kinect device. In our experiment, we only use the intensity data to evaluate our method.

B. Key frames extraction

For a specific dynamic hand gesture, there are various performance for different people even the same person. The hand pose and hand movement speed may changed every time. In Fig.4, there are three video sequences which both are the same gesture, where these hand gestures also are affected by the illumination. Temporal features is very important in dynamic gesture recognition. In order to effectively extract the temporal features of dynamic gestures, we first extract key frames from the video.

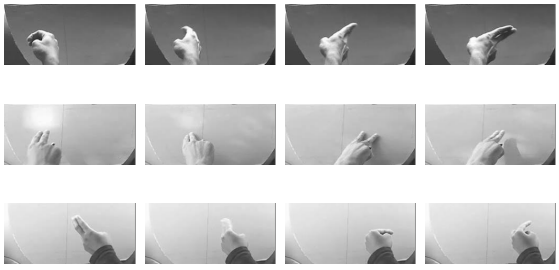


Fig. 4: Examples of dynamic hand gesture. These three gesture videos are with same label of "Swipe R" , which performed by different subjects. In addition, illumination has a great influence on gesture recognition.

Video key frames extraction is mainly used for video retrieval. It extracts the most representative some frames or one frame of the video to replace the entire video. There are various algorithms for video key frames extraction such as key frames extraction based on sampling, clustering and motion information. Sampling based key frames extraction is the most simple way which extracts a key frame with a fixed time interval. This method is used in many papers to extract key frames. However it is difficult to choose a reasonable interval, and many important temporal features will be lost if the time interval is long. In this paper, we utilize the motion information to extract the key frames. As known, optical flow can accurately extract the motion information of the target in a video. We use Horn-Schunck algorithm [45] to compute the optical flow of videos.

In Horn-Schunck algorithm, assuming that the image brightness is constant, $E(x, y, t)$ denotes the image intensity at the point (x, y) in the image plane at the time t. So, we have

$$E(x, y, t) = E(x + \Delta x, y + \Delta x, t + \Delta t) \tag{6}$$

Apply a first-order 3D Taylor expansion and obtain the following:

$$E(x+\Delta x, y+\Delta x, t+\Delta t) = E(x, y, t)+\frac{\partial E}{\partial x}dx+\frac{\partial E}{\partial y}dy+\frac{\partial E}{\partial t}dt \tag{7}$$

Using the brightness constancy, we have that

$$\frac{\partial E}{\partial x}dx + \frac{\partial E}{\partial y}dy + \frac{\partial E}{\partial t}dt = 0 \tag{8}$$

Divided by dt and let $E_x = \frac{\partial E}{\partial x}$, $E_y = \frac{\partial E}{\partial y}$, $E_t = \frac{\partial E}{\partial t}$,$u = \frac{dx}{dt}$,$v = \frac{dy}{dt}$, obtain the following:

$$E_x u + E_y v + E_t = 0 \tag{9}$$

Then, accroding to the Horn-Schunck energy equation, we get

$$u^{n+1} = u^n - E_x[E_x u^n + E_y v^n + E_t]/[\alpha^2 + E_x^2 + E_y^2] \quad (10)$$

$$v^{n+1} = v^n - E_x[E_x u^n + E_y v^n + E_t]/[\alpha^2 + E_x^2 + E_y^2] \quad (11)$$

where α is the weighting parameter and n is the iteration number. u and v are the horizontal component and vertical component of the optical flow. We define

$$T(t) = \sum_{i=1}^{h}\sum_{j=1}^{w} u(i,j,t) + \sum_{i=1}^{h}\sum_{j=1}^{w} v(i,j,t) \quad (12)$$

Where (x,y) and t are the position of pixel in optical flow image and time,respectively. $T(t)$ is the sum of all elements in u and v. We choose the top 12 of $T(t)$ corresponding the frames as the key frames as shown in Fig. 5.

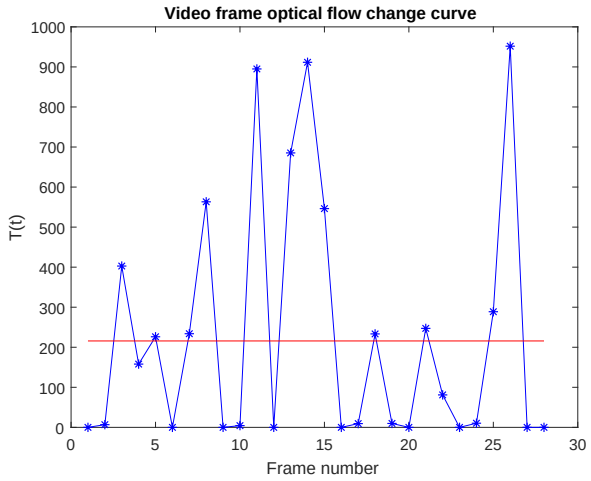


Fig. 5: Video frame optical flow change curve. Above the red line, there are 12 points corresponding 12 key frames.

C. Edge extraction

Because the dataset is collected under various illumination conditions, in order to reduce the influence of illumination and improve the robustness of the model, We extract the edges of the original frame sequence for training and testing our model. We adapt Canny algorithm [46] to extract edges because it has the following advantages. 1)Optimal Detection: this algorithm can identify as many actual edges as possible, and the probability of missing real edges and false detection of non-edges are relatively small. 2)Optimal location criterion: the position of the detected edge point is the closest to the actual edge point, or the degree of the detected edge deviating from the real edge of the object is the smallest because of the influence of noise.

D. Data augmentation

Due to VIVA dataset only contains 885 samples, if we use the raw data to train our mdCNN model, overfitting will be generated. The simplest and most common way to resist overfitting is to expand the original dataset artificially. Different from the image dataset, video dataset has two ways to enlarge data set: temporal and spatial augmentation. In this paper, a novel temporal data augmentation method is introduced. Combined with the traditional spatial augmentation, the overfitting is significantly reduced and the generalization ability of the model is improved. In the following, we introduced the temporal augmentation and spatial augmentation in detail.

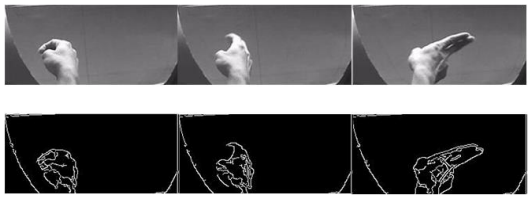


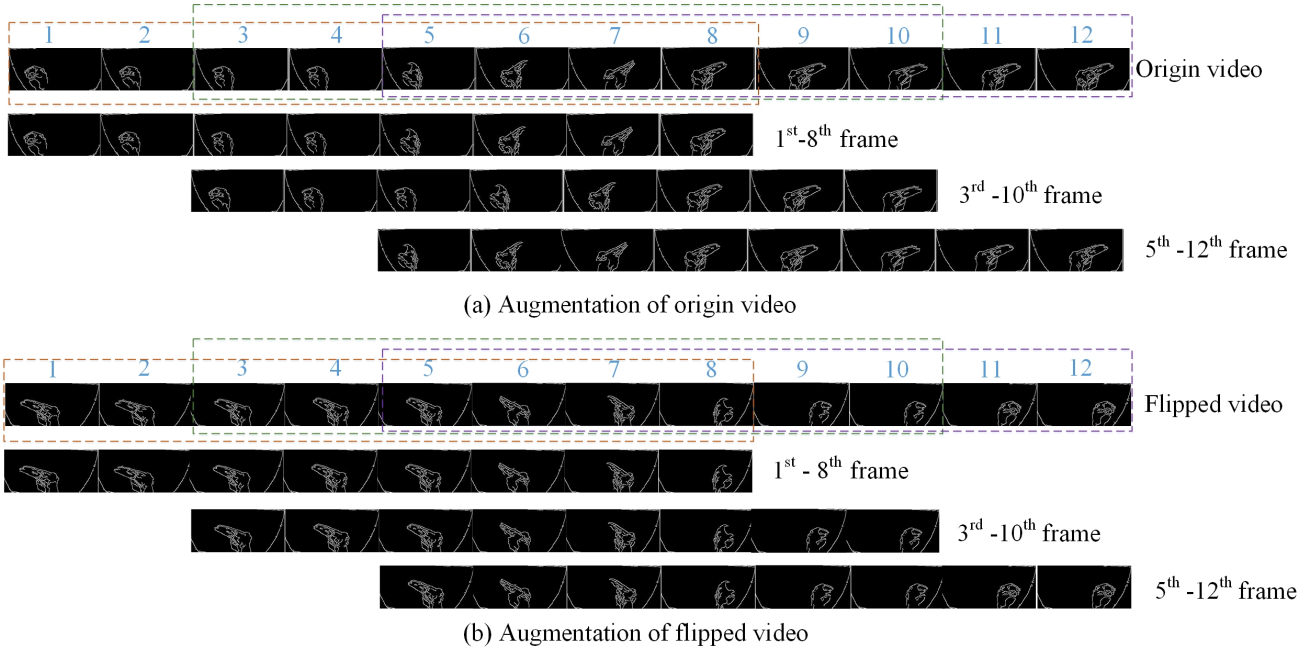
Fig. 6: Edge extraction. The first row is the original frame sequence, the second row is the edge frame sequence.

1) Temporal augmentation: A new data augmentation way which named temporal cropping is proposed in our work inspired by [21] to prevent overfitting. After extracted key frames, each video has 12 frames, then we temporally crop the video to 8 frames. In the train phase, we extract 8 frames from 1st frame, 3rd frame and 5th frame, then one sample will generate three training samples: 1st-8th frames, 3rd-10th frames and 5th-12th frames. In the test phase, we random extract 8 continuous frame sequences to test the model. In order to generate more training samples, we also do flip of some videos [35]. Different from the image horizontal flipping, there are two steps in video flipping. First, we reverse the ordering of the frame sequences and then flip each frame image horizontally. After these two operations, a new video is generated, and this video has the same label as the original video. We also temporal crop the flipped video with same way. By temporal data augmentation, the dataset was 6 times larger than origin dataset. Fig. 7 shows our temporal data cropping.

2) Spatial augmentation: Spatial augmentation is a traditional method to enlarge the dataset, which is widely applied in image classification. To further reduce the overfitting and improve the generalization ability of our mdCNN model, we also adapt the spatial augmentation. Motivated by some existing methods [2], our method comprise of three parts: 1.Gaussian smoothing. We adapt multi-scale gaussian smoothing with the variance σ are 1, 3 and 5 respectively. The larger the σ, the wider the frequency band of Gaussian filter, then the better the degree of smoothness; 2. Affine transform. We rotate the each frame with $\pm 10°$, and translate frames(± 5 pixels along the x axis, ± 10 pixels along y axis);3.Spatial crop. After the temporal data augmentation, the size of frames sequences is $115\times250\times8$, we down sample frame sequences in spatial to $57\times125\times8$. In the train phase, we extract five clips with size of $50\times100\times8$ from down sample frames(the four corner patches and center patches) used to training. But in the test time, a random clip with size of $50\times100\times8$, which extracted from test videos, are input to the model.

V. Experiments and Evaluation

After building the mdCNN architecture and processing the training data, we designed a series of experiments to evaluate



(a) Augmentation of origin video

(b) Augmentation of flipped video

Fig. 7: temporal data augmentation.

our model. In this section, we firstly in detail introduced our training process include the settings of convolution layer, pooling layer parameters and network optimize strategies. hen, we experiment to evaluate the effectiveness of the proposed data enhancement method. Finally, we compared our mdCNN with state-of-art methods, the classification results prove that our method is significantly better than other methods.

A. *Network parameters setting*

As shown in Fig.2, we propose two mdCNN architectures, and each architecture contains three branches(front-net,top-net,left-net). In addition to the different ways of fusion, the two architectures have the same parameter settings. Due to ReLU layer without parameters and all BN layers using the default parameter settings, we just care about the 3D convolutional layer and pooling layer. For the front-net, the input data with size of 50×100×8, the Conv1(first 3D convolution layer) consists of 4 convolution kernels with the size of 5×7×3. The Conv2 consists of 8 convolution kernels with size of 3×5×3. Convolution kernel size of the last two convolution layers are both 3×3×3, but Conv3 has 16 convolution kernels and Conv4 has 32 convolution kernels. In top-net and left-net, for convenience training in caffe, the input data need to be transformed to 8×100×50 and 50×8×100,respectively. The corresponding convolution layer parameters should also be changed for these two branches except the amount of convolution kernels. For top-net, from the first convolution layer to the forth convolution layer, the kernel size are 3×7×5, 3×5×3, 3×3×3, 3×3×3,respectively. For left-net, from the first convolution layer to the forth convolution layer, the kernel size are 5×3×7, 5×3×3, 3×3×3, 3×3×3,respectively. All 3D pooling layer of each sub-net are set to 2×2×2 except the first 3D pooling layer. The pool1 of front-net, top-net and left-net are set to 2×2×1, 2×1×2 and 1×2×2 respectively, which prevent the temporal information collapse in the early phase. In mdCNN-A, there are two fully connected layers after the fusion layer(concat layer), FC5 with 1024 neurons and FC6 with 2048 neurons, and then is connected to softmax layer to get final classification result. In mdCNN-B, each branch has two fully connected layers, FC5 with 512 neurons and FC6 with 1024 neurons, and the final classification results are obtained by averaging of three sub nets classification results. The drop out technique are also used in mdCNN, all FC5 with the dropout ratio of 0.7 and all FC6 with the dropout ratio 0.5.

B. *Training process*

We use mini-batch gradient descent [47] with momentum to optimize our models with a batch size of 24 examples,momentum of 0.9, and weight decay of 0.0005. The momentum gradient descent method reduces the fluctuation on the path to the minimum by accumulating the past gradient values and accelerates the convergence. The detailed weights update rule as follows:

$$v_{i+1} := 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \mathrm{d}w_i \tag{13}$$

$$w_{i+1} := w_i + v_{i+1} \tag{14}$$

where i is the i_{th} iteration, v is the momentum variable standing the accumulating of the past gradient, ϵ is the base learning rate and the $\mathrm{d}w_i$ is the gradient of loss function to w_i. The weight of 3D convolution layers are initialized by xavier method where weights are with uniform distribution between $[-\sqrt{\frac{6}{n_i+n_o}},+\sqrt{\frac{6}{n_i+n_o}}]$. n_i and n_o are the number of input and output neurons respectively. The biases of convolution layers and two fully connected layers were initialized to constant 1, and in softmax layer the biases are set to constant 0. The learning rate is initialized to 0.01 and then we adjust the learning rate manually. With all parameters setting completed, we start training our model to converge and start observing the error rate. If the error rate no longer falls with current learning rate, we divided the learning rate

by 10. The training was terminated after the learning rate reduced three times. The whole training process is about 80 epoches. It took about 0.5 hours to train our fusion network on an NVIDIA 1050ti GPU for a single fold of the leave-one-subject-out cross-validation on caffe [48].

C. Results

The VIVA dataset contains 8 subjects, so we evaluated our method using 8 folds cross-validation. We use 7 subjects to train our networks and leave 1 subject to test the network. This process was repeated 8 times and the final classification accuracy is the average of 8 test results. We compare the

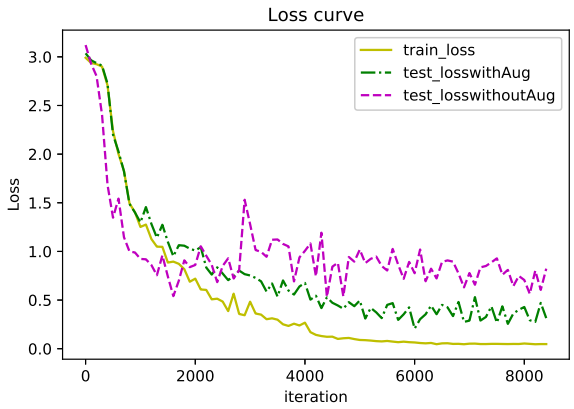


Fig. 8: Accuracy and loss curve. In the figure, $test\_losswithAug$ and $test\_losswithoutAug$ mean the test loss generated with and without using our data augmentation method respectively.

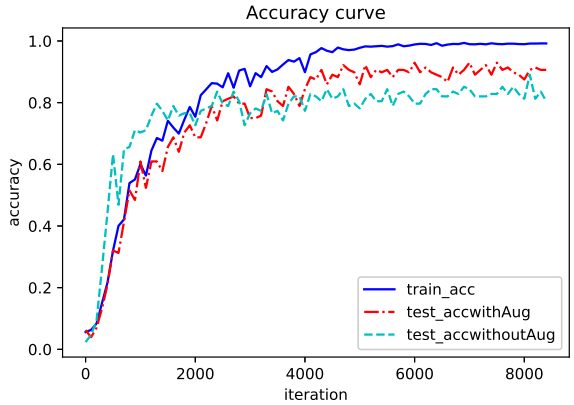


Fig. 9: Accuracy and loss curve. In the figure, $test\_accwithAug$ and $test\_accwithoutAug$ mean the test accuracy generated with and without using our data augmentation method,respectively.

loss results with using our data augmentation method or without using our data augmentation method by using our mdCNN-A model. In the Fig. 8, train loss(yellow solid line) is no longer change after about 7000 iterations and stopped to 0.1. In the first 2000 iterations, $test\_losswithoutAug$ drop faster than $test\_losswithAug$ because after the data augmentation dataset become larger and is difficulty to convergence. After the 2000 iterations, $test\_losswithAug$ continues fall down, but $test\_losswithoutAug$ changes small. Finally, $test\_losswithAug$ drops to near by 0.5 with a little fluctuant, $test\_losswithoutAug$ only drop to near by 0.8 and it had greater fluctuant. These experiments show that the data augmentation method that we proposed achieved good results in resisting of the overfitting. Beside comparing the loss, we also consider the influence of data augmentation on recognition accuracy. Similar with the loss curve, in the first the 2000 iterations, $test\_accwithoutAug$ performs better than $test\_accwithAug$, but finally, $test\_accwithAug$ increases to nearly by 0.88 better than $test\_accwithoutAug$ with 0.82, which is shown in Fig. 9.

We compare the recognition accuracy of each subject before and after data augmentation, experiment results are listed in Table I. From Table I, we can see that accuracy of each subject has different improvement after using the data augmentation. The average accuracy increased from 60.57% to 65.35%, with an increasement of 4.78%. Subject 1 achieved the greatest improvement, recognition accuracy is improved from 82.5% to 88.8%, achieving the increasement of 5.3%. Subject 7 get the smallest improvement from 63.6% to 67.3% with an increased of 3.7%. We also compare our proposed methods with other state of art methods. Note that in this paper we just use the RGB data not use the depth data.

TABLE II: The classification results of different methods

Method	Modality	Accuracy
Harris-3.5D	RGBD	36.4%
HOG3D	RGBD	44.6%
Dense Trajectories	RGBD	54%
HON4D	D	58.7%
HOG+HOG2	RGBD	64.5%
3DCNN	RGB	57%
mdCNN-A	RGB	**65.35%**
mdCNN-B	RGB	**64.76%**

From Table II, we can see that our two fusion methods performed better than past state-of-art methods, especially the mdCNN-A which get the best accuracy with 65.35%. The accuracy of mdCNN-B is 64.76% that is lower than mdCNN-A. We infer the main reason is that these two model adapted different fusion ways. For mdCNN-A, three sub nets fuse after forth convolution layer in feature-level, which synthesize spatial-temporal features from three directions. However, the result of mdCNN-B was only the average accuracy of the three sub nets. Therefore, the former is more explanatory and theoretical. Before our method, the best result is 64.5% get by HOG+HOG2 method with RGBD data, which is lower 0.95% than mdCNN-A. In [35], they also proposed a fusion net that contained two resolution sub nets, high resolution net and low resolution net, with the accuracy of 57% far below our 65.35%. Besides, the input of our model is a clip with 8 frames, but in their model the input is 32 frames. Therefore, our model converged faster and process more data than it in the same time.

D. Visualization

Convolutional neural networks are end-to-end learning models. The transmission of the input data or feature maps

TABLE I: Accuracy of each subject

Subject	1	2	3	4	5	6	7	8	average
AccWithoutAug	82.5%	62.8%	43.6%	67.1%	60.7%	52.5%	63.6%	51.8%	60.57%
AccWithAug	88.8%	67.1%	48%	72.2%	65.5%	57.6%	67.3%	56.3%	65.35%

between the each layer is implicit to us. The visualization of convolutional neural networks provides a way for us to understand the learning process of neural networks. Visualization is divided into two types: One is visualization of convolution kernels, which can help us understand what features each convolution kernel extracts. Another is visualization of feature maps, which helps us understand the output of the input data after each convolutional layer. It is difficult to show the visualization of 3D convolution kernels, so in this paper we only visualize a part of the output feature maps of each 3D convolution layer of front-net. There are 4

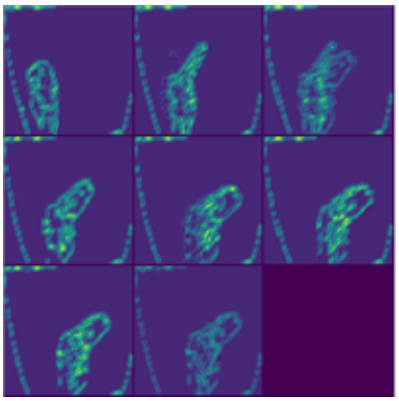


Fig. 10: One of the output feature maps of Conv1.

convolution kernels in Conv1, which can be output 4 feature maps. The Fig. 10 only shows one of the output features of Conv1 where 8 images correspond to input images sequence. In the third image, we can see the motion information of the adjacent image. For a clearer observation, we deliberately visualized the 4 feature images of the third image in Fig. 11, where 4 images correspond to 4 convolution kernels. Fig. 12

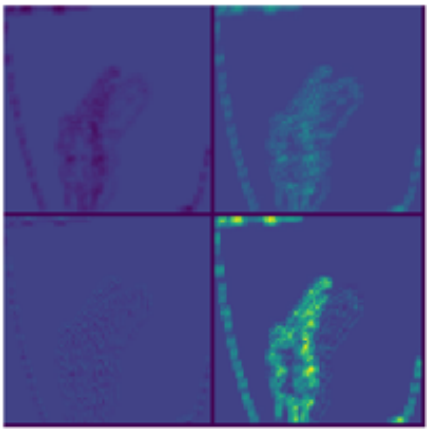


Fig. 11: Four feature maps of third image.

shows the post convolution images of Conv2, darker areas represent motion information. It also has 8 images, because in the first 3D pooling layer, the depth of kernel is one, so the length of the image sequence will not be halved. In Fig. 13,(a) and (b) show the visualization of Conv3

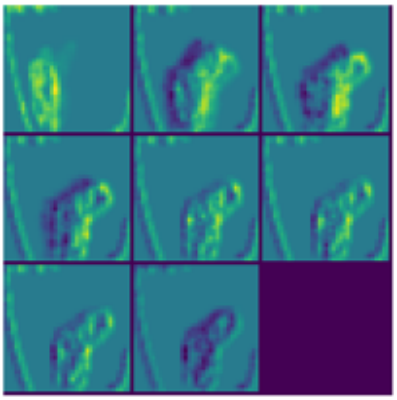


Fig. 12: One of the output feature maps of Conv2

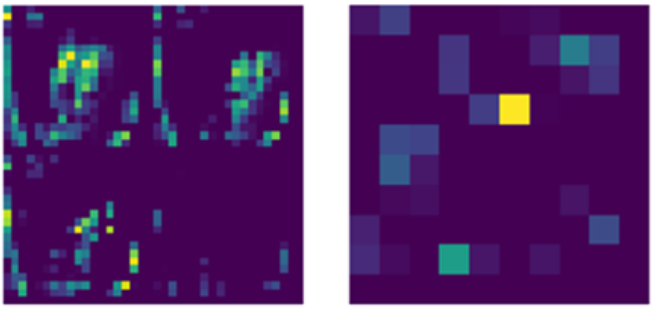


Fig. 13: One of the output feature maps of Conv3(a) and Conv4(b)

and Conv4 feature maps respectively. We can also see the gesture area dimly from fig.13(a). But in fig.13(b), the image becomes abstract because of the size of image is small.

E. Confusion matrix

In order to better analyze the classification of each category in dynamic gestures, this paper calculates the confusion matrix of the classification results. According observation of the confusion matrix from Fig. 14, we can see that "Pinch" has the lowest recognition accuracy of 57%, and "Swipe +" achieves the highest recognition accuracy of 86%."Swipe R", "Swipe L", "Swipe D" and "Swipe U" are easily misclassified as "Scoll R", "Scoll L", "Scoll D" and "Scoll U". The most serious is "Swipe D", which has a 23% probability of being classified as "Scoll D" and a 15% probability of being classified as "Swipe D". We show these two gestures in Fig. 15. The first row is "Swipe D", and the

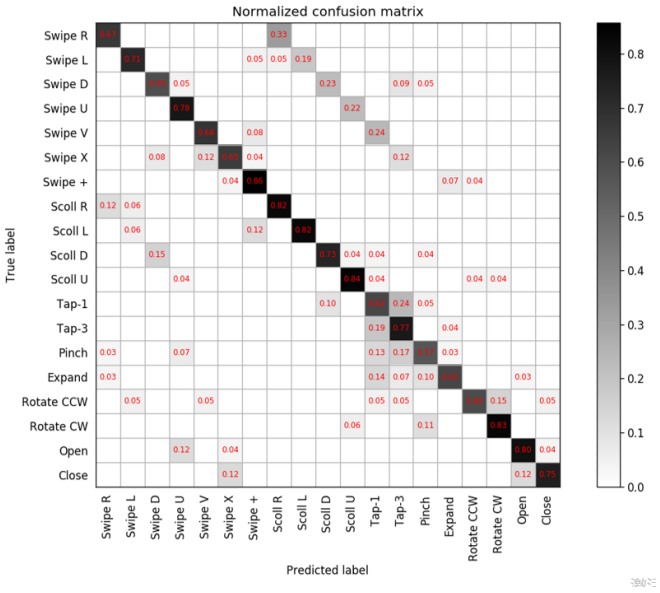


Fig. 14: Confusion matrix

Fig. 15: ”Swipe D” and ”Scoll D”

second row is ”Scoll D”. In ”Swipe D”, two fingers first go down and then turn into fists, However, in ”Scoll D”, two fingers go up and then down. So the two kinds of gestures are very similar, which leads to the classification error of neural network.

VI. Conclusion

In this paper, we designed mdCNN model which can extract complete spatial-temporal features from multi-direction by using 3D convolutional neural network to recognize the dynamic hand gesture in car and we proved that effectiveness on VIVA challenge dataset. Some video preprocessing work was did. We utilized the optical flow to extract the key frames abandoning the redundancy frames, which can accelerate net convergence and improve the accuracy rate. To avoid overfitting, a new data augmentation method was proposed, and the experiment result showed that our method get the good performance and improved the classification accuracy. However, there are still some points for improvement in our work in future works: 1. A effective mechanism to resist overfitting. Data augmentation is a convenience way, but overfitting still exists in training, whether we can solve this problem from the perspective of CNN-self. 2. More effective ways to extract spatial-temporal features. The accuracy of mdCNN-A is only 0.59% higher than that of mdCNN-B, so more effective fusion ways of multi-direction features or more complete spatial-temporal features need to be employed. 3. Whether existing more effective video key frame extraction method. Although the optical flow method achieves very good results in key frame extraction, it consumes computational resources. In the future works, we focus on solving the above problems.

References

[1] H. Zhuang, M. Yang, Z. Cui, and Q. Zheng, “A method for static hand gesture recognition based on non-negative matrix factorization and compressive sensing,” *IAENG International Journal of Computer Science*, vol. 44, no. 1, pp. 52–59, 2017.

[2] V. I. Pavlovic, R. Sharma, and T. S. Huang, “Visual interpretation of hand gestures for human-computer interaction: a review,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 677–695, Jul 1997.

[3] S. Mitra and T. Acharya, “Gesture recognition: A survey,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 3, pp. 311–324, May 2007.

[4] B. Ionescu, D. Coquin, P. Lambert, and V. Buzuloiu, “Dynamic hand gesture recognition using the skeleton of the hand,” *Eurasip Journal on Advances in Signal Processing*, vol. 2005, no. 13, pp. 1–9, 2005.

[5] S. B. Wang, A. Quattoni, L. Morency, and D. Demirdjian, “Hidden conditional random fields for gesture recognition,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1521–1527.

[6] P. Trindade, J. Lobo, and J. P. Barreto, “Hand gesture recognition using color and depth images enhanced with hand angular pose data,” in *Multisensor Fusion and Integration for Intelligent Systems*, 2012, pp. 71–76.

[7] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, June 2005, pp. 886–893 vol. 1.

[8] A. Klaser, M. Marszalek, and C. Schmid, “A Spatio-Temporal Descriptor Based on 3D-Gradients,” in *BMVC 2008 - 19th British Machine Vision Conference*, M. Everingham, C. Needham, and R. Fraile, Eds. Leeds, United Kingdom: British Machine Vision Association, Sep. 2008, pp. 275:1–10. [Online]. Available: https://hal.inria.fr/inria-00514853

[9] E. Ohn-Bar and M. M. Trivedi, “Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 6, pp. 2368–2377, Dec 2014.

[10] N. H. Dardas and N. D. Georganas, “Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques,” *IEEE Transactions on Instrumentation & Measurement*, vol. 60, no. 11, pp. 3592–3607, 2011.

[11] S. Nowozin and J. Shotton, “Action points: A representation for low-latency online human action recognition,” 7 J J Thomson Ave, CB30FB Cambridge, UK, Tech. Rep., July 2012.

[12] T. Starner, J. Weaver, and A. Pentland, *Real-time American sign language recognition using desk and wearable computer based video*. IEEE Computer Society, 1998.

[13] S. B. Wang, A. Quattoni, L. P. Morency, D. Demirdjian, and T. Darrell, “Hidden conditional random fields for gesture recognition,” in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*, vol. 2, 2006, pp. 1521–1527.

[14] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, “Dense trajectories and motion boundary descriptors for action recognition,” *International Journal of Computer Vision*, vol. 103, no. 1, pp. 60–79, May 2013. [Online]. Available: https://hal.inria.fr/hal-00803241

[15] M. Zobl, R. Nieschulz, M. Geiger, M. Lang, and G. Rigoll, “Gesture components for natural interaction with in-car devices,” in *Gesture-Based Communication in Human-Computer Interaction, International Gesture Workshop, Gw 2003, Genova, Italy, April 15-17, 2003, Selected Revised Papers*, 2004, pp. 448–459.

[16] F. Althoff, R. Lindl, and L. Walchsh?usl, “Robust multimodal hand- and head gesture recognition for controlling automotive infotainment systems,” 2008.

[17] F. Parada-Loira, E. Gonzalez-Agulla, and J. L. Alba-Castro, “Hand gestures to control infotainment equipment in cars,” in *Intelligent Vehicles Symposium Proceedings*, 2014, pp. 1–6.

[18] Q. Zhang, M. Yang, K. Kpalma, Q. Zheng, and X. Zhang, “Segmentation of hand posture against complex backgrounds based on saliency and skin colour detection.” *IAENG International Journal of Computer Science*, vol. 45, no. 3, pp. 435–444, 2018.

[19] Q. Zheng, X. Tian, S. Liu, M. Yang, H. Wang, and J. Yang, “Static hand gesture recognition based on gaussian mixture model and partial differential equation.” *IAENG International Journal of Computer Science*, vol. 45, no. 4, pp. 596–583, 2018.

[20] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, p. 436, 2015.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc.,

2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[22] Q. Zheng, M. Yang, Q. Zhang, and J. Yang, "A bilinear multi-scale convolutional neural network for fine-grained object classification." *IAENG International Journal of Computer Science*, vol. 45, no. 2, pp. 340–352, 2018.

[23] Q. Zheng, M. Yang, Q. Zhang, and X. Zhang, "Fine-grained image classification based on the combination of artificial features and deep convolutional activation features," in *2017 IEEE/CIC International Conference on Communications in China (ICCC)*, Oct 2017, pp. 1–6.

[24] R. Girshick, "Fast r-cnn," *Computer Science*, 2015.

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," in *International Conference on Neural Information Processing Systems*, 2015, pp. 91–99.

[26] Q. Zheng, M. Yang, Q. Zhang, X. Zhang, and J. Yang, "An end-to-end image retrieval system based on gravitational field deep learning," in *2017 International Conference on Computer Systems, Electronics and Control (ICCSEC)*, Dec 2017, pp. 936–940.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," pp. 1–9, 2014.

[29] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," *ArXiv e-prints*, Dec. 2014.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[32] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," vol. 1, no. 4, pp. 568–576, 2014.

[33] L. Alvarez, J. Weickert, and J. Snchez, "Reliable estimation of dense optical flow fields with large displacements." *International Journal of Computer Vision*, vol. 39, no. 1, pp. 41–56, 2000.

[34] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool, "Temporal segment networks: Towards good practices for deep action recognition," *Acm Transactions on Information Systems*, vol. 22, no. 1, pp. 20–36, 2016.

[35] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, "Hand gesture recognition with 3d convolutional neural networks," in *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2015, pp. 1–7.

[36] S. Y. Cheng and M. M. Trivedi, *Vision-based infotainment user determination by hand recognition for driver assistance*. IEEE Press, 2010.

[37] G. Lefebvre, S. Berlemont, F. Mamalet, and C. Garcia, "Blstm-rnn based 3d gesture classification," in *International Conference on Artificial Neural Networks and Machine Learning, ICANN*, 2013, pp. 381–388.

[38] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *Computer Science*, 2015.

[39] X. Shi, Z. Chen, H. Wang, W. C. Woo, W. C. Woo, and W. C. Woo, "Convolutional lstm network: a machine learning approach for precipitation nowcasting," in *International Conference on Neural Information Processing Systems*, 2015, pp. 802–810.

[40] P. Molchanov, S. Gupta, K. Kim, and K. Pulli, "Multi-sensor system for driver's hand-gesture recognition," in *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, vol. 1, May 2015, pp. 1–8.

[41] N. Neverova, C. Wolf, G. W. Taylor, and F. Nebout, "Multi-scale deep learning for gesture detection and localization," in *Computer Vision - ECCV 2014 Workshops*, L. Agapito, M. M. Bronstein, and C. Rother, Eds. Cham: Springer International Publishing, 2015, pp. 474–490.

[42] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Journal of Machine Learning Research - Proceedings Track*, vol. 9, pp. 249–256, 01 2010.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[44] Q. Zheng, M. Yang, J. Yang, Q. Zhang, and X. Zhang, "Improvement of generalization ability of deep cnn via implicit regularization in two-stage training process," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2018.

[45] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1, pp. 185 – 203, 1981. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0004370281900242

[46] J. Canny, *A Computational Approach to Edge Detection*. IEEE Computer Society, 1986.

[47] Y. Lcun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[48] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

Jie Li was born in Weifang, Shandong, China in 1994. He received his B.S. degree from Shandong Agriculture University in 2015 and begin work for a M.S. degree in Shandong University in 2016. His research direction is computer vison and machine learning.