Proceedings of the 2019 IEEE 16th International Conference on Networking, Sensing and Control, May 9-11, 2019, Banff, Alberta, Canada

Dynamic Hand Gesture Recognition Based on 3D Convolutional Neural Network Models

Wenjin Zhang
Department of Computer Science and Software Engineering
Monmouth University
West Long Branch, USA
s1260807@monmouth.edu

Jiacun Wang
Department of Computer Science and Software Engineering
Monmouth University
West Long Branch, USA
jwang@monmouth.edu

***Abstract*—Hand gesture is a natural communication method which could be used to create a more convenient interface for human-robot interaction. In this study, we use the simplest laptop camera as an input sensor. We designed a 3D hand gesture recognition model. The model is trained with the Jester dataset. After being trained about one day in a MacBook Pro (i5 2.3GHz), the model reached an average accuracy of 90%. We built a web application that implements the hand gesture recognition system and provides the recognition service to users.**

Keywords—Hand gesture recognition, Deep learning, neural network, 3D CNN, Convolution neural network

I. Introduction

The definition of human gestures is any state or motion of the special part of the human body, especially the face and hand. Among them, it is most effective for hand gesture to replace the spoken language and communicate with each other. This is why the deaf usually use the hand gesture to express his means, for example, American sign language is one of the hand gesture communication method. Therefore, hand gesture could be a natural communication method which is usually used by the deaf.

With the development of information technology and computer science, people cannot live well without a computer. However, traditional input devices, like a mouse, keyboard and so on, can no longer meet users' interactive requirements which impede the process of natural user interface because there is always a strong barrier between the computer and user through the traditional technology. Thus, it is becoming more and more important to be able to interact with the computer naturally in the area of Human-Computer Interaction[1].

For creating more convenient interfaces (HRI) for human-robot interaction researchers have proposed many methods to recognize the hand gesture including Non-vision-based approaches and vision-based approach, which is classified by input device[2][3]. Non-vision-based approaches usually use wearable equipment with some sensors (optical or mechanical), such as accelerometer and gyroscope, which could translate the hand motion into an electrical signal. The theoretical basis of these approaches is that the different hand gestures have different electrical signals. Although these methods have high recognition, they have a lot of limitations because people have to wear these sensors all the time. Except that, these methods cannot perform very well to recognize the static hand gesture because the sensors couldn't capture any signal when the hand is still[4]. On the contrary, vision-based approaches choose single, multiple cameras or precise body motion senses such as Leap Motion, Microsoft Kinect and so on. In theory, the hand gestures could be recognized as long as they are visible to the cameras. In addition, vision-based approaches could deal with static hand gestures. Vision-based approaches are more practical than Non-vison based approaches but their implementation will be complex because of huge input data. Nowadays, most of the researches about hand gesture recognition focus on the vision-based approaches.

This study applies a deep learning method to recognize hand gestures. We design a 3D convolution neural network model and attempt to use the Jester V1.0 hand gesture dataset to train this model. According to the result of the training experiment, it got an average accuracy of 90%.

The organization of this paper is as follows: Section II reviews the related work including Convolution neural network and Action Recognition. Section III introduce the related the concept of convolution neural network. Section VI designs the 3D-CNN mode for hand gesture recognition. Section V introduces the hand gesture dataset and the result of training. Section VI build a web application implementing hand gesture recognition and provides this service to users. Section VII summarizes this paper and proposes future works.

II. Related Work

A. Convolution Neural Networks

In recent years, researchers pay more and more attention to applying the Convolutional Neural Networks (CNNs) in the area of computer vision. The CNNs have achieved great success in the Recognition, Motion analysis, Scene reconstruction, and Image Restoration. LeCun et al. proposed the standard convolutional neural networks called 'LeNet-5' and applied this network into recognizing the handwritten digits achieving an average accuracy of 99.3% in the MNIST Dataset in 1998[5]. Based on LeCun, Simard et al., Bernhard Schölkopf et al. and Ciregan et al. optimized LeCun's method improved the accuracy to 99.6%, 99.61%, and 99.64% respectively[6][7][8]. Kyeongryeol et al. proposed a low-power convolutional neural network to develop a (CNN)-based face recognition system[9]. Kaiming et al. presented a residual learning framework to improve the convolution neural networks' performance of image recognition who won the first price in the area of detecting images, locating images, detecting COCO and segmenting COCO in the ILSVRC&COCO 2015 Competitions 1[10]. According to the

978-1-7281-0084- 5/19/$31.00 ©2019 IEEE

literature review, it is obvious that the convolution neural network is the best choice for computer vision at present.

B. *Action Recognition*

Dynamic hand gesture recognition can be seen as a branch of action recognition, whose task is to identify the different actions by requiring the context from a whole video of an action rather than from each frame. This means that action recognition needs to identify the different state of action from each 2D frame where the action may or may not performed and then synthesize the obtained state to determine what this action is.

Although the deep learning method has achieved great success in the area of image classification in the past years, the progress of video classification research is slow. The reasons why this research is difficult are as follows.

1) *Standard Benchmark*

UCF101 and Sports1M datasets have been recognized as the standard benchmark by the research for a long time, but there are also some problems with this benchmark. About UCF101, although the number of data could be able to meet the requirement, the actual diversity of this dataset is much lesser because of the high spatial correlation. In addition, another problem is that both of them have the same theme, sports. It is difficult to prove that the method works well in these themes could be generalized into other tasks.

2) *Computational Cost*

When we want to recognize an action, the input data must be a video or a sequence of images from a camera which means huge computation cost. According to Tran, Du et al., they spent 3 to 4 days to train a deep learning model on UCF101 and It is unbelievable to spend about 2 months on Sports-1M data for training[11].

III. Designing 3D-CNN model for hand gestures recognition

This chapter will briefly introduce the concept of neural networks and design the 3D convolutional neural network model for hand gesture classification.

A. *Artificial Neural Networks*

1) *Network Structure*

An artificial neural network is a simulation of the biological neural network and works like a human brain because it is inspired by the biological nervous network in the brain. Actually, the neural network itself is a framework for the different machine learning algorithm which could process complex input data, rather than a specifical algorithm. The artificial neural networks could learn the experience of performing some tasks from a lot of examples which is usually called training data. After that, the neural network could complete the task without any man-made rules. For example, LeCun et al. designed the Le-net neural network model to learn how to recognize the handwritten digits '0-9' from the train data called 'MNIST' which have been manually labeled accurately and achieved an accuracy of 99.3% in the test dataset of 'MNIST'. They did this task without any prior knowledge about handwritten digits, for example, that and the digits '0' is an ellipse and the digit '1' is a long vertical.

An artificial neural network consists of layers of artificial neural and each layer is a set of neural. As Figure 1 shows, we assume that there are m+1 input signals from x0 to x_m and the weights are from w_0 to w_m. Particularly, the first input x_0 is usually assigned value +1, which is a clever way to use w_{k0} as the bias because it makes $w_{k0} = b_k$. Except for x0, the remaining x input signal to the neural: from x_1 to x_m is the actual inputs. Therefore, the output of the kth neuron could be represented as:

$$y_k = f(\sum_{j=0}^{m} w_{kj} x_j)$$

In this formula, f is the activation function which is really important for artificial neural networks. If a neural network isn't applied with the activation function, this network is a simple regression model and cannot learn and model the complicated data like images, audio, and videos. The activation function could introduce the non-linear properties to artificial neural networks. In addition, it is helpful to perform back propagation optimization method like gradient descent because one feature of an activation function is that it should be differentiable everywhere which could ensure that the partial derivatives always exist. Nowadays, there are three types of activation functions including Sigmoid, Hyperbolic tangent (Tanh) and Rectified liner units (Relu) [12]. Almost all deep learning models use Relu as the activation function, so this study also chooses it.

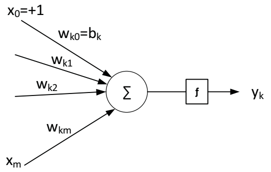

Figure 1: Basic structure of artificial neural

The most popular and classic artificial neural network is the multilayer perceptron (MLP) which is a class of feedforward artificial neural network. An MLP network has an input layer, an output layer and at least one hidden layer. In Figure 2, there are two hidden layers and each neuron connects to each of the following layer neurons but that in the same layer cannot connect to each other. Because of that, it is also called a fully connected network.

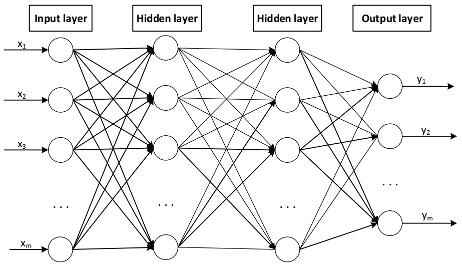

Figure 2: Architecture of the MLP network

B. 2D Convolution Neural Networks

Fully connected neural networks like multilayer perceptron have a huge drawback that the depth of network could not be very large because of huge computation cost for fully connecting, which means that it will never reach the complexity of the human brain. This is why the convolution neural network was proposed which uses the sparse connectivity and shared weights approaches to reduce the parameters.

2D convolution neural networks is usually knowns as convolution neural network (CNN or ConvNet) which is a variation of multilayer perceptron and commonly applied to analyzing visual imagery. The architecture of convolution is built by three types of layers: Convolution Layer, Pooling Layer and Fully connected Layer and repeat modules indicate that these modules usually repeat more than one time (M>0, 0<N and usually N<3) in Figure 3. For example, the architecture of classic convolution neural, Lenet, is Input→ {Conv→Pooling}*2 → FC *3 → which performs very well in the ImageNet [13][14].

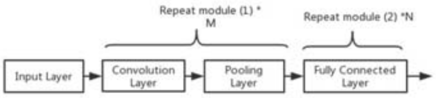

FIGURE 3: THE COMMON ARCHITECTURE OF CONVOLUTION NEURAL NETWORK （Note: Activation functions are included by Convolution Layers.）

Convolution layer is the key element of a convolution network, whose parameters consist fo a set of learnable kernels. During forwarding in the convolution layer, the filter (kernel) will slide over the input matrix as a fixed stride and the weight of a filter should multiply by the input data in Figure 4. Neural of convolution layer just connect to a small region of the output of previous neural called 'receptive field', whose size is the same as that of a filter. Although its receptive field is small for avoiding the fully connected and reducing the learnable parameters, it extends through the full depth of the input volume.

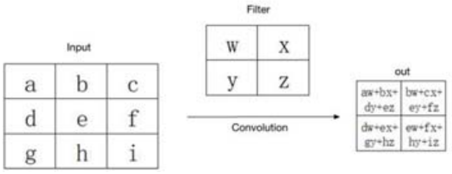

FIGURE 4. AN EXAMPLE OF CONVOLUTION.

C. 3D Convolution Neural Networks

3D Convolution neural network can be seen as a variant of 2D convolution neural network extending 2dimension filter into 3 dimensions. This 3D filter shall slide in 3 directions to extract low-level features and its output's shape is a 3dimension space like a cuboid[15].

According to the practice, 2D Convolution neural networks have achieved great performance in processing the image data. In fact, filters of 2D Convolution neural networks extract the features from 2D data, like a single image. Assuming that the order of frame is the third dimension, the feature of 3D data like videos, could be extracted. However, the image processed by 2D convolution neural network is a grayscale image, 2D data and the frames of a video is RGB data, 3D data because they have three channels. There are two methods to deal with this problem. One of them is to convert each frame to grayscale images and then merge grayscale images into a matrix as the 3D convolution neural network input. This method is easy to implement and reduce the computation, but it misses some features. Another method is to separate RGB image into 3 types of grayscale images and then merge them into 3 matrixes as the order of frames (different type is in the different matrix). In the end, the input of the 3D convolution neural network shall have three channels. This method could keep all the features, but it needs some computation.

In this paper, we designed this 3D-CNN architecture for hand gesture recognition which is described in Figure 5. The input of this architecture is 18 84*84*3 images. When the input data flow into this network, it will go through four times 3D convolution and 3D Max pooling operation and each output after convolution should be applied with 'Relu' activation function. And then, it will flow into 'Fully Connection' and Dropout layers designed for overfitting problem. The last layer of this architecture is also 'Fully Connection' which could adjust the neural network output that the probability for each gesture class.

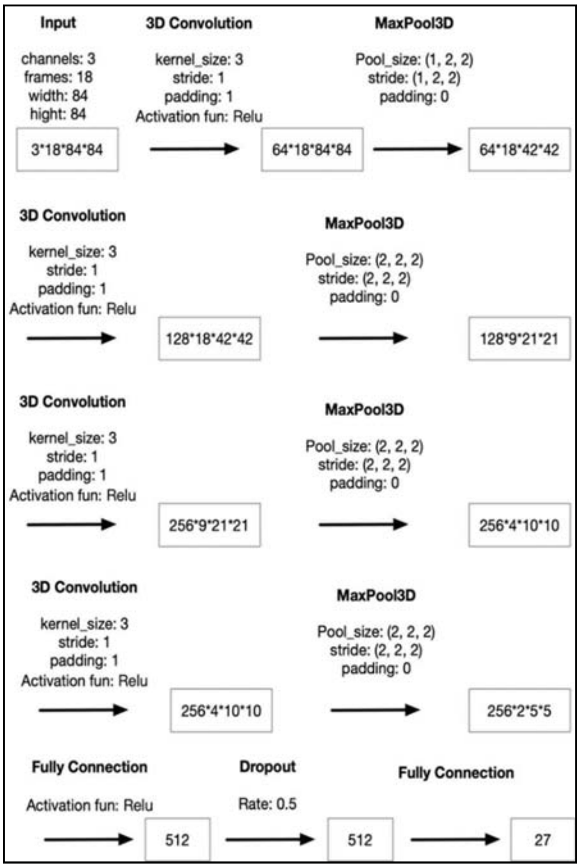

FIGURE 5: 3D-CNN ARCHITECTURE

IV. EXPERIMENT

In this section, we trained our model in an environment as follows: MacBook Pro (I5 2.3GHz), mac OS Mojave, Pycharm, Python 3.6, Pytorch 1.0, pillow 5.3. Based on this model, we build a web application implementing this model to provide the recognition service to users.

A. *System Architecture*

As Figure 6 shows, the architect of the system includes 3 components: Camera, Preprocessing Module and 3D-CNN Model. A simple camera is used to obtain the video stream. The responsibility of the preprocessing module is to change the form of a video stream and normalize the data. Because obviously, the input that our model needs are a series of images, this module shall extract the images from the video stream at 12 frames per seconds and then normalize these data for 3D-CNN recognition module. When the feature vector flow through this module, we could get the prediction result.

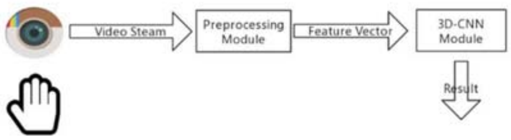

FIGURE 6: SYSTEM ARCHITECTURE

B. *Hand Gesture Dataset*

TABLE 1. LIST OF HAND GESTURES.

Class Index	Gesture Description
0	Swiping Down
1	Swiping Left
2	Swiping Right
3	Swiping Up
4	Thumb Down
5	Thumb Up
6	Turning Hand Clockwise
7	Turning Hand Counterclockwise
8	Zooming In With Full Hand
9	Zooming In With Two Fingers
10	Zooming Our With Full Hand
11	Zooming Out With Two Fingers
12	Stop Sign
13	Sliding Two Figures Up
14	Sliding Two Figures Right
15	Sliding Two Figures Left
16	Sliding Two Figures Down
17	Shaking Hand
18	Rolling Hand Forward
19	Rolling Hand Backward
20	Pushing Two Figures Away
21	Pushing Hand Away
22	Pulling Two Figures In
23	Pulling Hand In
24	No gesture
25	Drumming Fingers
26	Doing other things

In this study, we choose the 20BN-JESTER dataset, which was collected and organized by Twenty BN, to train and test our model. This dataset is a large collection of label video clips that show humans hand gestures collected by laptop camera or webcam. All the hand gestures were created by a lot of crowd worker rather than very few people. In addition, they performed this gesture in the very complex background such as rotating fan, bright bulb, moving cat and so on. Because of that, this dataset is a good choice to train the robust machine learning models for hand gesture recognition. There are 27 types of hand gestures listed in Table 1.

There are 148,092 videos in this dataset: 118,562 for training, 14,787 for validation and 14,743 for testing, which is provided as TGZ archive. Each video was converted into JPG images at 12 frames per seconds whose name start at 00001.jpg. So, the archive fold has 148092 directories which contain these images.

C. *Training Process*

The process of network training is to learn the experience from the training dataset including preparing phase and training phase. As Figure 7 shows, the preparing phase is used to prepare the training sample and initializing the parameters of the model. Most of the time, the training dataset could feed into the neural network model directly which means that data preprocessing, like Normalization (improving convergence speed), should be applied to the original data.

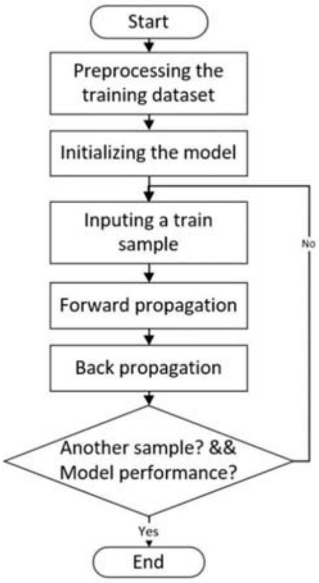

FIGURE 7: MODEL TRAINING PROCESS

The training phase is a cycle process including inputting a training sample, forward propagation and back propagation that will keep doing until the model achieves great performance or there are no more training data. The back propagation plays the most important role in the training phase whose function is to back propagate error and update the parameters. In essence, the performance of the model is measured by the loss function and the goal of the training process could be converted to minimizing the loss between the prediction value and the desired output. For example, one of the loss functions is squared loss function, which is a fairly straightforward loss function, although it usually doesn't perform very well. It is defined as:

$$L(n) = \frac{1}{2}\sum_{k \in C}(y_k(n) - y'_k(n))^2 .$$

In this formula, C is the set of neural output and y_j and y'_j is the true and prediction value of k-th neural, respectively. In this study, we choose CrossEntropyLoss loss function. Nowadays, the parameters are updated by using gradient descent methods such as Stochastic Gradient Descent (SGD), Mini-Batch Stochastic Gradient Descent, Momentum and Adagrad. Nowadays, all of these optimizers have been developed by machine learning framework like tensorflow or pytorch.

D. Data Preprocessing

During the previous research, data processing plays a very important role in the machine learning area. As this paper mentioned before, the key element of the machine learning algorithm is the optimization process of gradient descent which could search on the loss function surface to find the best parameter for the minimum loss. How about unnormalized data? For example, we can imagine a scenario in which there is a simple network with two input. Among them, one input value, a, ranges from 0 to 10, but another input value, b, varies from 0 to 1. Although the best parameters could be found in theory, unfortunately, most of the time the convergence of model is going to be very slow or even not at all because gradient descent only focuses on some variable of input data in Figure 8, When the input data on different scales. On the contrary, the network could learn the best parameters faster on the same scale input data in Figure 9.

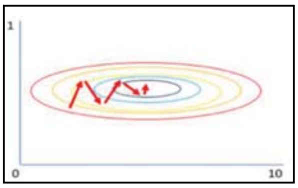

FIGURE 8: GRADIENT DESCENT ON DIFFERENT SCALE INPUT

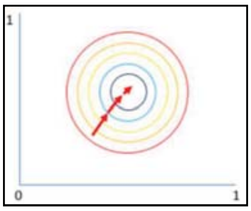

FIGURE 9: GRADIENT DESCENT ON THE SAME SCALE INPUT

In the field of machine learning, the process of converting the data of different scales into that of the small scale is called as normalizing. The standardization method is one of the popular normalizing methods, defined as:

$$x' = \frac{x - \overline{x}}{\sigma}$$

, where x is the original input data, $\overline{x}$ is the mean of all the input data, and σ is the standard deviation of it. In this study, we use this method to normalize the input data. Another preprocessing is to extract the images from the video steam at 12 frames per seconds because the input of our model is a series of images.

E. Training Result

In this study, we record three variables to evaluate our models:

1) loss: the difference between the prediction value and expected value. This study chose the CrossEntropyLoss function.

2) top1_accuracy: the percentage of the one with the highest probability is the expected answer.

3) top3_accuracy: the percentage of the expected answer is one of the model's 3 highest probability answer.

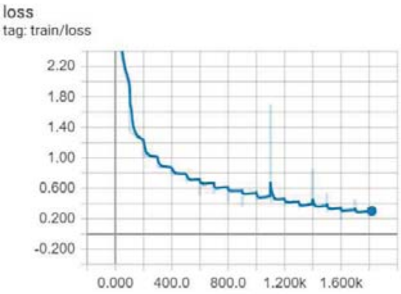

FIGURE 10: THE TRAINING LOSS

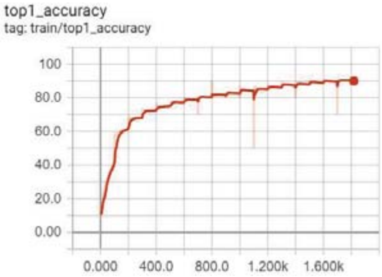

FIGURE 11: TOP 1 ACCURACY: THE PERCENTAGE OF THE ONE WITH THE HIGHEST PROBABILITY IS THE EXPECTED ANSWER. X–AXIS: NUMBER OF TRAINING BATCH; Y-AXIS: TOP 1 ACCURACY.

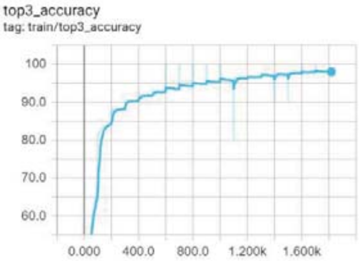

FIGURE 12: TOP 3 ACCURACY: THE PERCENTAGE OF THE EXPECTED ANSWER IS ONE OF THE MODEL'S 3 HIGHEST PROBABILITY ANSWER. X-AXIS: NUMBER OF TRAINING BATCH; Y-AXIS: TOP 3 ACCURACY.

During the training phase, the loss has been on a downward trend, which indicates that the training process is effective in Figure 10. After 1600 batch training, it was stable at around 0.3 indicated that the training has finished. It cost about 3 days to train this model in the MacBook Pro (i5 2.3GHz). As Figure 11 and Figure 12 show, the top1_accuracy and top3_accuracy rose very fast at the beginning and then fluctuated around 90%, 98% respectively. The testing accuracy is stable at about 87% after 25 epochs. Another contrast experiment using 64*64 image sequence and adding an extra convolutional layer has been done. It cost about 2 days 4 hours to train 1600 batch data but the average accuracy was stable at 82%.

This accuracy could not be considered as the best one, some other methods has an average accuracy of 96%. For example, Oman Köpüklü proposed motion fused frames method that using the combination of optical flow frames and static images as the input data to increase the recognition accuracy. Although the accuracy performance of his method is perfect, it cost more times than the method proposed in this study. We have tried to convert the optical flow from the jester dataset and it cost about 8 hours in my mac. What's more, when we applied this our method in practice, the performance is great.

F. *Dynamic testing*

During the testing phase, we choose a common classroom as our testing background. The complex background and the bright light were our testing challenges in Figure 13. After all the hand gestures were tested, the result indicated that the model could accurately recognize hand gestures in the complex background and the robustness of our model is proved.

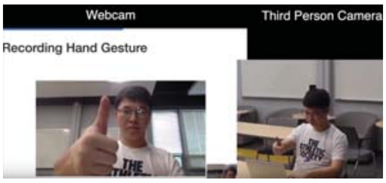

FIGURE 13: THE RUNNING SCREENSHOT OF THE SYSTEM

V. CONCLUSIONS

The 3D Convolution neural network model has been designed for hand gesture recognition, which includes 4 convolution layers, 2 fully connection layers and 1 dropout layers. We spent about 3 days to train this model using the jester dataset in the MacBook Pro and this model achieved an accuracy of 90%. And then we developed a web application integrating this model to provide hand gesture recognition service through Restful API. For testing this service, a demo web application has been developed to drive the camera. This application was tested in a bedroom and proved that it could recognize the hand gesture effectively.

Future work of this study includes:

1. The accuracy of this study has some space for increasing by using some motion extraction algorithm or a more complex network;
2. The distance between human and camera has limitation. If a human sits far away from the camera, it won't work.

REFERENCES

[1] J. S. Sonkusare, N. B. Chopade, R. Sor and S. L. Tade, "A Review on Hand Gesture Recognition System," 2015 International Conference on Computing Communication Control and Automation, Pune, 2015, pp. 790-794.

[2] B. K. Chakraborty, D. Sarma, M. K. Bhuyan and K. F. MacDorman, "Review of constraints on vision-based gesture recognition for human–computer interaction," in IET Computer Vision, vol. 12, no. 1, pp. 3-15, 2 2018.

[3] S. Berman and H. Stern, "Sensors for Gesture Recognition Systems," in *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 3, pp. 277-290, May 2012.

[4] Sushmita Mitra, Tinku Acharya, Gesture recognition: a survey, IEEE Trans. Syst. Man Cyber – C: Appl. Rev. 37 (3) (1999) 311–324.

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

[6] M. Shilman, Zile Wei, Sashi Raghupathy, P. Simard and D. Jones, "Discerning structure from freeform handwritten notes," Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings., Edinburgh, UK, 2003, pp. 60-65 vol.1.

[7] Bernhard Schölkopf, John Platt, Thomas Hofmann, "Efficient Learning of Sparse Representations with an Energy-Based Model," in Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference, MITP, 2007, pp.

[8] D. Ciregan, U. Meier and J. Schmidhuber, "Multi-column deep neural networks for image classification," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, 2012, pp. 3642-3649.

[9] K. Bong, S. Choi, C. Kim and H. Yoo, "Low-Power Convolutional Neural Network Processor for a Face-Recognition System," in *IEEE Micro*, vol. 37, no. 6, pp. 30-38, November/December 2017.

[10] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 770-778.

[11] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In ICCV, 2015. 1, 2, 4, 5, 6, 7, 8

[12] Nair, Vinod; Hinton, Geoffrey E. (2010), "Rectified Linear Units Improve Restricted Boltzmann Machines", 27th International Conference on International Conference on Machine Learning, ICML'10, USA: Omnipress, pp. 807–814, ISBN 9781605589077

[13] A. Ardakani, C. Condo, M. Ahmadi and W. J. Gross, "An Architecture to Accelerate Convolution in Deep Neural Networks," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 4, pp. 1349-1362, April 2018.

[14] J. Garland and D. Gregg, "Low Complexity Multiply Accumulate Unit for Weight-Sharing Convolutional Neural Networks," in *IEEE Computer Architecture Letters*, vol. 16, no. 2, pp. 132-135, 1 July-Dec. 2017.

[15] S. Ji, W. Xu, M. Yang and K. Yu, "3D Convolutional Neural Networks for Human Action Recognition," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221-231, Jan. 2013.