**Dynamic Hand Gesture Recognition Based on Short-Term Sampling Neural Networks**

**by**

**Wenjin Zhang**

**Friday, 12/13, 10:00 am - 11:00 am**

**A thesis presented to Monmouth University**

**in partial fulfillment of the requirements for the degree of**

**Master of Science**

**In**

**Software Engineering**

**Monmouth University**

Approved by:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor Jiacun Wang (Supervisor)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor (2nd Reader)



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor Daniela Rosca (Chair)

# Abstract

Hand gestures are a natural communication method that can be used for human-robot interaction. Currently, 3D convolutional neural networks (CNN) are the most popular deep learning networks used to train and recognize hand gestures from video inputs. However, 3D CNNs suffer from intensive computation cost. In this study, a short-term sample approach is proposed to train a hand gesture recognition model. In this new approach, the whole video frames are segmented into several groups. A frame is randomly selected from each group as an input for model training. Then, the features fetched from the samples with CNN is fused with features mined from the technique of optical flow. The approach reduced the complexity of model training and meanwhile improve gesture recognition accuracy. Besides, an approach that, based on a given frame of video, generates a new video frame to make an object in the frame appear distant away is developed to test out the robustness of the model. For experiment, Jester hand gesture dataset is chosen to train the model. Experiment results show that the new approach achieves an average accuracy of 95.41% and 95.39% on the Jester dataset and the “zoomed-out” Jester dataset, respectively.

# Keywords

Deep learning, hand gesture recognition, optical flow, convolutional neural networks.

# Acknowledgments

I would like to express my deepest appreciation to my advisor, Jay Wang who is always being my strong support over the past two years. I couldn’t get this big project perfectly done without his guide and encouragement. He gave me so much inspiration both in research and personal life.

I am extremely grateful to Professor Yu, who was always showing her kindness to me, I practiced my oral English a lot by discussing this project with Prof. Yu. During this summer research, I learned a lot from her about how to design a service-oriented architecture (SOA) system and business database.

I would also like to thank Monmouth University for giving us the best equipment and academic atmosphere for studying.

I was very lucky to have an opportunity to work at Vydia Inc. as a software engineering intern for three months. Over there I worked with awesome Software Engineers and learning a lot of practice knowledge. I also want to thank all of them for their help and friendship.

# List of Figures

Figure 1: The common architecture of convolutional neural network

Figure 2. An example of the convolutional operation

Figure 3: Fusion idea for the action recognition

Figure 4: Two stream architecture for the action recognition

Figure 5: Dense optical flow of three pedestrians walking in different directions

Figure 6: The process of the short-term sampling neural network

Figure 7: System Architecture

Figure 8: Some examples of hand gesture videos from Jester Dataset.

Figure 9: The whole process of extending a video of the dataset

Figure 10: An example of extending a frame

Figure 11: The comparison between the original dataset and the “zoomed-out” Jester dataset

Figure 12: Gradient descent on different scale input

Figure 13: Gradient descent on the same scale input

Figure 14: The change of loss

Figure 15: Result on the validation set of Jester Dataset with the different group number

Figure 16: Result on the validation set of the “zoomed-out” Jester Dataset with different group numbers

Figure 17: the testing screenshot

Contents

[Abstract 2](#_Toc27046390)

[Keywords 2](#_Toc27046391)

[Acknowledgments 3](#_Toc27046392)

[List of Figures 4](#_Toc27046393)

[Chapter 1 - Introduction 7](#_Toc27046394)

[1.1 Hand Gesture Recognition 7](#_Toc27046395)

[1.2 Application Areas 8](#_Toc27046396)

[1.3 Thesis organization 8](#_Toc27046397)

[Chapter 2 – Background knowledge 9](#_Toc27046398)

[2.1 Convolutional neural network 9](#_Toc27046399)

[2.2 Action Recognition 11](#_Toc27046400)

[2.3 Optical Flow 13](#_Toc27046401)

[Chapter 3 – Short-term Sampling Neural Network 15](#_Toc27046402)

[3.1 The reasons for this design 15](#_Toc27046403)

[3.2 The Network Architecture 18](#_Toc27046404)

[3.3 Implementation Details 20](#_Toc27046405)

[Chapter 4 – Experiments 21](#_Toc27046406)

[4.1 System overview 21](#_Toc27046407)

[4.2 Dataset 22](#_Toc27046408)

[4.2.1 Jester dataset 22](#_Toc27046409)

[4.2.2 the Zoomed-out Jester dataset 24](#_Toc27046410)

[4.3 Data preprocessing 27](#_Toc27046411)

[4.4 Training Details 28](#_Toc27046412)

[4.4.1 Environments 29](#_Toc27046413)

[4.4.2 Training configuration 29](#_Toc27046414)

[4.5 Results and Analysis 30](#_Toc27046415)

[4.5.1 Change of loss 30](#_Toc27046416)

[4.5.2 Accuracy performance 30](#_Toc27046417)

[4.5.3 Comparison with other approaches 32](#_Toc27046418)

[4.6 System testing 33](#_Toc27046419)

[Chapter 5 – Conclusion 34](#_Toc27046420)

[Reference 35](#_Toc27046421)

# Chapter 1 - Introduction

## 1.1 Hand Gesture Recognition

Human gestures are defined as any state of motion of the special part of a human body, especially the face and hand. Among them, it is most effective for hand gestures to replace the spoken language and communicate with each other. This is why the deaf usually use the hand gesture to express his thoughts, 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. 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 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].

To create more convenient interfaces for human-robot interaction researchers have proposed many methods to recognize hand gestures including non-vision-based approaches and vision-based approaches, which is classified by the 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 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. On the contrary, vision-based approaches choose single, multiple cameras or precise body motion senses such as Leap Motion, Microsoft Kinect and so on. [4][5][6] 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 focuses on vision-based approaches.

## 1.2 Application Areas

Hand gesture recognition can be applied to in virtual environment control, sign language translation and robot remote control.

## 1.3 Thesis organization

The rest of the thesis is organized as follows: Chapter 2 introduces background knowledge about hand gesture recognition; Chapter 3 introduces our approach to recognize the hand gesture; Chapter 4 describes the experiment details and results; At the end, chapter 5 describes the conclusion about this study and the future work about hand gesture recognition.

# Chapter 2 – Background knowledge

This section will introduce the convolutional neural network and some other technique related to hand gesture recognition.

## 2.1 Convolutional neural network

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[7]. 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[8][9][10]. Kyeongryeol et al. proposed a low-power convolutional neural network to develop a (CNN)-based face recognition system[11]. Kaiming et al. presented a residual learning framework to improve the convolutional 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[12]. According to the literature review, it is obvious that the convolutional neural network is the best choice for computer vision at present.

The convolutional neural networks were proposed using the sparse connectivity and shared weights approaches to reduce the parameters of fully connected neural network. Fully connected neural networks like multilayer perceptrons 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.

A 2D convolutional neural network is usually known as the convolutional neural network (CNN or ConvNet) which is a variation of multilayer perceptrons and commonly applied to analyzing visual imagery. The architecture of convolutional is built by three types of layers: convolutional layer, pooling layer and fully connected layer. These layers may repeat more than one time (M>0, 0<N and usually N<3), as shown in Figure 1, where activation functions are included in the convolutional Layers. For example, the architecture of classic convolutional neural, Lenet, is Input🡪 {Conv🡪Pooling} \* 2 🡪 FC \*3 🡪 which performs very well in the ImageNet[13][14].

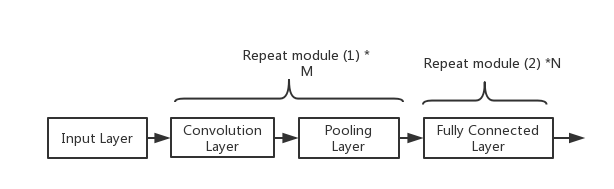


Figure 1: The common architecture of convolutional neural network

A Convolutional layer is the key element of a convolutional network, whose parameters consist fo a set of learnable kernels. During forwarding in the convolutional 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 2. Neural of convolutional layer just connects to a small region of the output of the previous nodes 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 2. An example of the convolutional operation

## 2.2 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 the standard benchmark and huge computational cost. 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. 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 [15].

Recently, all the approaches of action recognition in the area of deep learning fall into two categories: single stream network and two stream network. The major difference between these two categories is the design choice around combining spatiotemporal information. In 2014, Karpathy et al. explore multiple ways to use the 2D pre-trained convolutional neural network to fuse the temporal information from consecutive frames[16]. As Figure 3 shows, all the inputs of the structure are the consecutive frames. Fusing information from all frames at the end by using the single architecture is a single frame structure. Late fusion structure uses one more net sharing parameters to fetch feature and fuse the predictions at the last stage. Early fusion is the same as a single frame structure, but it combines the information from more than 10 frames in the first layer. Slow Fusion is a combination of Late Fusion and Early Fusion. Although the authors did a lot of experiments, the results of them are worse than that of the hand-crafted feature-based algorithm.

Considering the difficulty of the deep neural network to learn movement feature, Simmoyan and Zisserman suggest learning the movement feature in the form of stacked optical flow[17]. This architecture is shown in Figure 4 and it has two separate neural networks. Among them, one is designed for spatial context from original frames and another one is used to learn motion context from optical flow pictures. The author trains the two networks separately and fusing them by using support vector machine. This approach improves the performance of the single stream method.

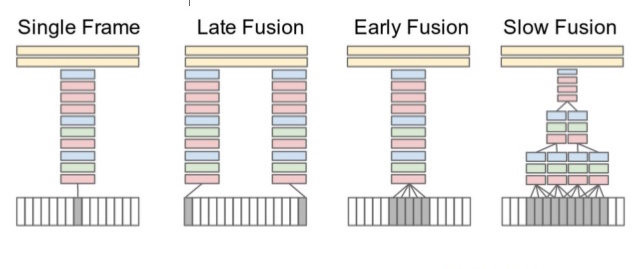


Figure 3: Fusion idea for the action recognition

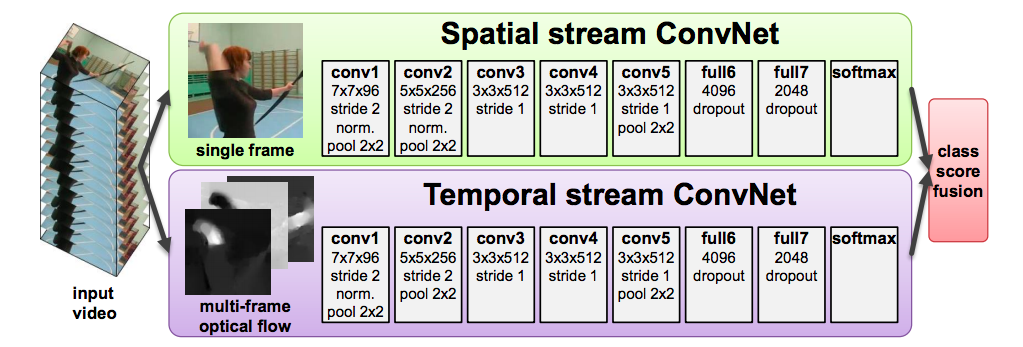


Figure 4: Two stream architecture for the action recognition

## 2.3 Optical Flow

Optical flow is the pattern of apparent motion of image objects between two consecutive frames due to the movement of object or camera[18][19]. In the 1940s, James J. Gibson first introduce the concept of optical flow[20]. After that, the followers of Gibson have done more research about that and have further demonstrated the importance of the optical flow for detecting the movement[21]. Optical flow is a great approach to analyse video. Recently, motion estimation has become a major aspect of optical flow research.

Assuming that we take an image frame at time , and then we take another image frame at time after . The aim of the optical flow approach is to calculate the motion between two frames. If a pixel of the first frame at location is defined , this pixel should be at location , where , and are the displacements between two frames, after . Due to brightness constancy, . Imaging that the is very small, which means that the movement is very small and according to Taylor series, . Combining the previous equations, we can get an equation, . Both sides of this equation are divided by , , which results in

,

where , and are the derivatives of the image at x, y and t, respectively, and , are the x and y components of the optical flow or velocity of . In the above equation called Optical Flow equation, we could know , and . However, it is common sense that we could calculate two unknown variables with only one equation. Thus, several methods are proposed to handle this problem, like Lucas-Kanade, Horn-Schunck[22], Farnebäck[23]. There are two types of optical flow: sparse and dense optical flow. Sparse optical flow gives some specific pixels in the frame while the dense optical flow gives the flow vectors of the whole frame. Although dense optical flow costs much more computation, it is more useful to detect the movement.

In this study, we choose Farnebäck’s method to calculate dense optical flow which is one of the most popular implementations. Firstly, this method uses quadratic polynomials to approximates the windows of image frames through polynomial expansion transform. And then, it defines a method to estimate displace areas form polynomial expansion coefficients through observing how the polynomial transforms under motion. We could get the dense optical flow after a series of refinements [23]. The result of applying the dense optical flow algorithm is shown in Figure 5.

  
Figure 5: Dense optical flow of three pedestrians walking in different directions.

# Chapter 3 – Short-term Sampling Neural Network

In this section, we give a detailed description of our short-term sampling neural network, STSNN. Firstly, we explain the reasons for this design. Secondly, we discuss the detailed architecture of the short-term sampling neural network. At the end, we talk about the implementation details about our approach.

## 3.1 The reasons for this design

This design based on the observations of the disadvantage of applying the 3D convolutional neural network (3D CNN) model on video recognition and the advantage of the recurrent neural network model (RNN) on processing the sequence model. The motivation of this design is to increase the performance by reducing the input data and training parameters of the model.

According to the previous study, the 3D CNN model was a special design for video recognition. The 3D CNN extends 2 dimension filters into 3 dimensions. The extra dimension was used to handle the time sequence relationship between frames. Although it is effective to handle video data, it also introduces a huge computation cost. For example, we use 4 convolutional layers and 2 fully connected layers to build a 3D CNN model for recognizing the hand gesture. The parameters of this model are shown in Table 1, it costs about 80MB memory for each video data during the forwarding process and trains about 146M parameters. This disadvantage causes two problems: it is difficult and takes a long time to train this model; it is hard to deploy an embedded operating system because of so many parameters[24].

Table 1 The parameters of the 3D convolutional neural network.

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer** | **Data shape** | **Memory** | **Parameters** |
| INPUT | 3\*18\*84\*84 | 381K | 0 |
| CONV-1 | 64\*18\*84\*84 | 8.1M | (3 \* 3 \* 3) \* 64 \* 18 = 31,104 |
| POOL-1 | 64\*18\*42\*42 | 2.1M | 0 |
| CONV-2 | 128\*18\*42\*42 | 4.1M | (3 \* 3 \* 64) \* 128 \* 18 = 1,327,104 |
| POOL-2 | 128\*18\*21\*21 | 1.0M | 0 |
| CONV-3 | 256\*18\*21\*21 | 2.032M | (3\* 3 \* 128) \* 256 \* 18 = 5,308,416 |
| POOL-3 | 256\*18\*10\*10 | 461K | 0 |
| CONV-4 | 512\*18\*10\*10 | 921K | (3 \* 3 \* 256) \* 512 \* 18 = 21,233,664 |
| POOL-4 | 512\*18\*5\*5 | 230K | 0 |
| FC1 | 1\*1\*512 | 512 | 512 \* 18 \* 5 \* 5 \* 512 = 117,964,800 |
| FC2 | 1\*1\*27 | 27 | 512 \* 27 = 13,824 |

In order to develop a model from large volume information with less input, some experts sample the frames from the original video at a fixed rate and capture the features from each frame, and then use the RNN model to fuse the features as the output. But, the fewer video frames lead to the missing of spatiotemporal context across frames, and thus it is also important to enhance the movement information by combining the optical flow pictures. It is a common sense and important discovery that although an action video densely records multiple frames, the content changes relatively slowly. Based on this discovery, a short-term sampling approach could be useful by dividing the consecutive video frames into a fixed number of group. And then, we pick one sample which combines the optical flow data from each group and gather them as the neural network input. The benefit of this approach is to sample frames for the whole video and fix the number of input samples regardless of the duration of the videos that we are handling.

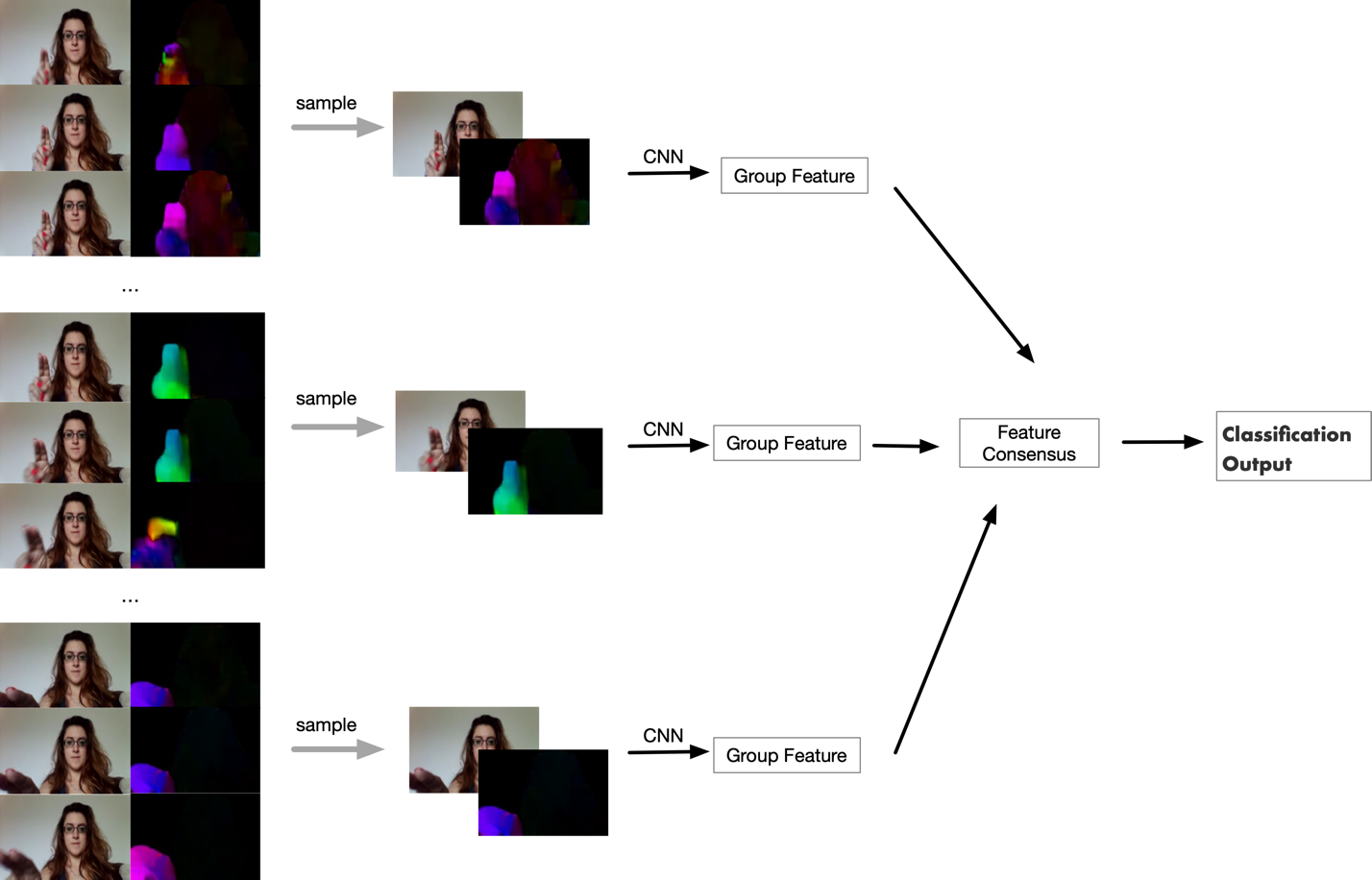


Figure 6:  The process of the short-term sampling neural network. It divides the input video frames and optical frames to n groups and the representative sampling is randomly selected from each group. And then, the sample is generated by combining the original frame and the optical flow frame. The CNN architecture is used to fetch the group feature. The different group features flow into the feature consensus module and are fused for classification.

## 3.2 The Network Architecture

In this section, we give a detailed description of the network architecture of STSNN for dynamic hand gesture recognition.

As Figure 6 shows, generally, the input video frames are divided into N groups {G1, G2, …, GN}. The frame number in each group is defined as , where is the total number of frames in the whole frames. Then, we sample group and get sampling data which is a combination of the randomly selected original frame and optical flow picture. After that, the formula of the short-term sampling neural network is defined as follows:

.

where, is the prediction value of the short-term sampling neural network, is features that the CNN network fetches from the sampling and is the parameters of CNN filters which are shared on different groups. The function is used to combines all the output from the CNN network to achieve a consensus data of class possibility. In the end, this consensus data should be passed to which is a hypothesis function. After that, we could get the final probability of each classification. In this study, we choose the softmax function as the hypothesis function which defined as , where n is the number of types and is the probabilities of target class over all the target classes. The benefit of using softmax function is to ensure that the range of the probabilities is from 0 to 1, and the sum of all the probabilities is equals to 1. Thus, the return value of our network is the probabilities of each class and the predicted class has the highest probability.

The most important component in this network is the consensus function , as it not only have the ability to fuse the partial-level prediction probabilities into video-level prediction but also is differentiable or have sub-gradient because it should support gradient descent optimization algorithm for backpropagation. What’s more, the work of Limin et al. show that the gradient-based optimization method could update the parameters from the loss of the whole video rather than just one group[25]. Assuming that the consensus function output is , and our loss function is standard categorical cross-entropy loss, the final loss function, , should be as follows:

,

Where, is the actual value and is the probabilities for label, is the number of hand gesture classes, and is the dimension of . According to the gradient descent parameter updating formulate, the change of the parameter in one training iteration is related to the gradients of the loss with respect to the parameter needing to be trained, as follows:

,

where is the number of groups. According to this formula, we know that the process of parameters training requires the consensus data from the whole video, which means that our approach could update the parameters by learning from the whole frames rather than only one group. Comparing with the previous densely sampled study, like 3D CNN, the computational cost goes down sharply. For example, assuming that the number of groups, N, is 5, there will be only 5 sampling images flow into our model. Moreover, all the CNN filter is 2 dimensions rather than 3 dimensions.

## 3.3 Implementation Details

Considering that the dataset for hand gesture recognition, Jester, is relatively small which could lead to the model fall into over-fitting and the training performance, our goal is to improve the training performance and avoid the over-fitting condition.

To improve the training performance and save training time and computation cost, transfer learning plays a very important role recently. According to the work of Pan et al., an image classification model designed for one domain may apply to another area because their data may have the same feature and distribution[26]. A good practice of transfer learning is to extract features by using the CNN network. Thus, we choose the pre-trained model Inception architecture with Batch Normalization, Inception v2, because it not only has a good balance between efficiency and accuracy but also the batch normalization could reduce the over-fitting which leads to better test performance through better generalization[27]. The Dropout layer is another technique to reduce the overfitting problem. One extra dropout layer was added after the pooling layer in Inception and its radio was set as 0.5[28].

# Chapter 4 – Experiments

In this section, we first introduce the hand gesture dataset and how we did to enhance it to improve the robustness of the trained model. Then, we discuss the train details about our approach, such as optimization function, learning rate and so on. At the end, we compare the performance of our method with other approaches posted on the Jester learning board.

## 4.1 System overview

As Figure 7 shows, the architecture of the system includes 3 components: input device, preprocessing module and recognition 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 second and then normalize these data for the recognition module. When the data flow into this module, we could get the prediction result. After that, we could create a web service to share the recognition result with other applications to achieve the purpose of control.

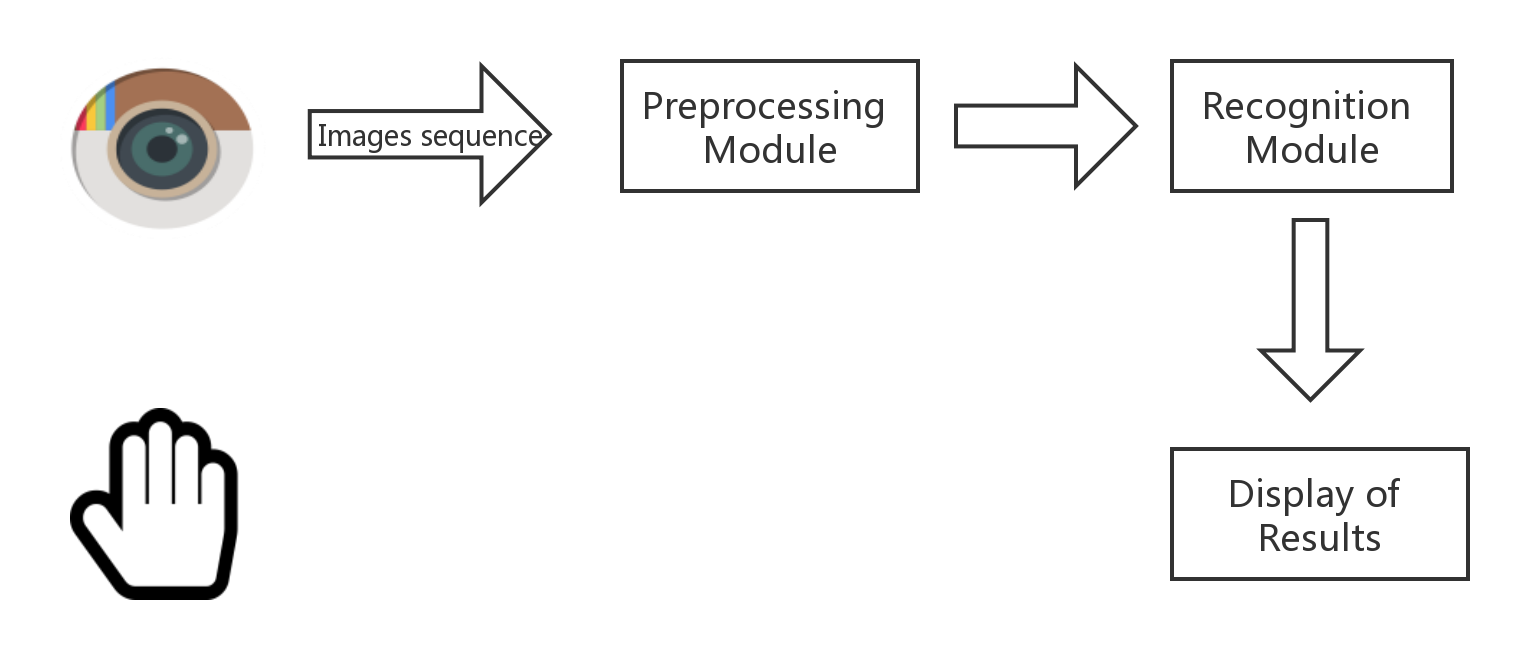


Figure 7: System Architecture

## 4.2 Dataset

### 4.2.1 Jester dataset

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 labelled video clips that show humans hand gestures collected by using the laptop camera or webcam. All the hand gestures were created by a lot of crowd workers rather than very few people. In addition, they performed this gesture in a very complex background such as a rotating fan, bright bulb, moving cat and so on. As Figure 8 shows, there is a large variance of the appearance of people, and complex ambient occlusion and background scenes. Because of that, this dataset is a good choice to train the robust machine learning models for hand gesture recognition[29]. 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 second whose name starts at 00001.jpg. So, the archive fold has 148092 directories that contain these images. There are 27 types of hand gestures listed in Table 2. Among them, 25 of them are gesture classes and 2 of them, ‘No gesture’ and ‘Doing other things’, are special classes that could not be recognized as any hand gestures. ‘No gesture’ means that a user sits or stands still without any movement and ‘Doing other things’ is a collection of various activities exception 25 hands gesture included in the dataset. Thus, based on this dataset, it is possible to handle these two exceptions and apply our approach to some specific applications.

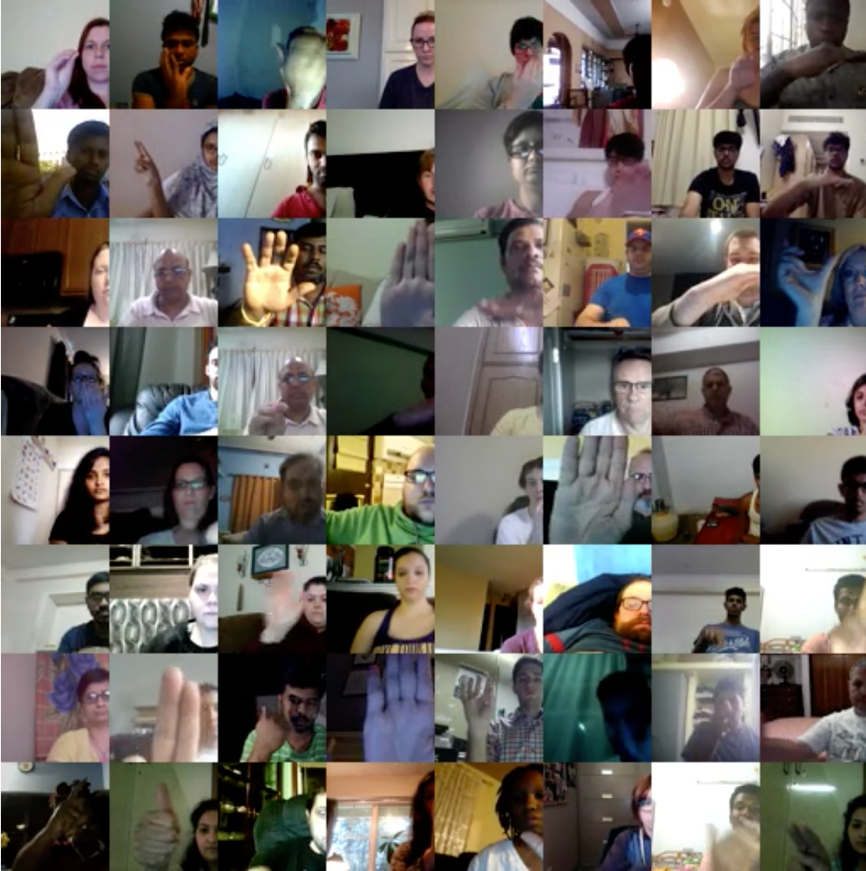


Figure 8: Some examples of hand gesture videos from Jester dataset.

Table 2. List of hand gestures.

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

### 4.2.2 the Zoomed-out Jester dataset

The data acquisition process of the Jester dataset is that all the actors recorded the hand gestures in front of webcams, and sitting close to the webcams. That’s the limitation of this dataset for a specific application. When we want to apply this model to an application, we cannot assume that users always do the hand gesture close to the webcam or camera. That’s our motivation to enhance this dataset, which is, we “zoomed out” video frames in the dataset.

In order to zoom out the Jester dataset, we should consider one discovery that adjacent pixels have strong similarities, especially the background pixels, like white walls or sofa. Thus, our approach is to duplicate boundary pixels of the frames. It may introduce a problem whether it will affect the original image movement or not. So, we should make sure that the boundary pixels which should be duplicated do not contain any movement. As mentioned before, the optical flow algorithm could detect the movement between different frames. According to our experiment, we found that the movement of 31.4% of video occurs at the boundaries of the video. Thus, we couldn’t duplicate the movement part. Thus, our approach should make sure the duplicated pixels should be a background which means it won’t affect the whole video movement. A great solution is to extend the whole video based on only one frame of this video, which means we could make sure that all the extend part is background in the whole video. In order to extend the dataset so that the distance between actors and camera varies within a range, we set a maximum number of extended pixels, , and then we randomly generate a number of extended pixels for each video which should in the range of . The whole process of extending video frames is shown in Figure 9. In Figure 10, Frame (b) is generated by filling the surrounding area of the original frame(a) with the boundary pixels of the original frame and then the “zoomed-out” frame is achieved by resizing frame (b). Comparing the original frames and ‘zoomed-out’ frames, this approach effectively increases the distance between the actor and camera in Figure 11.

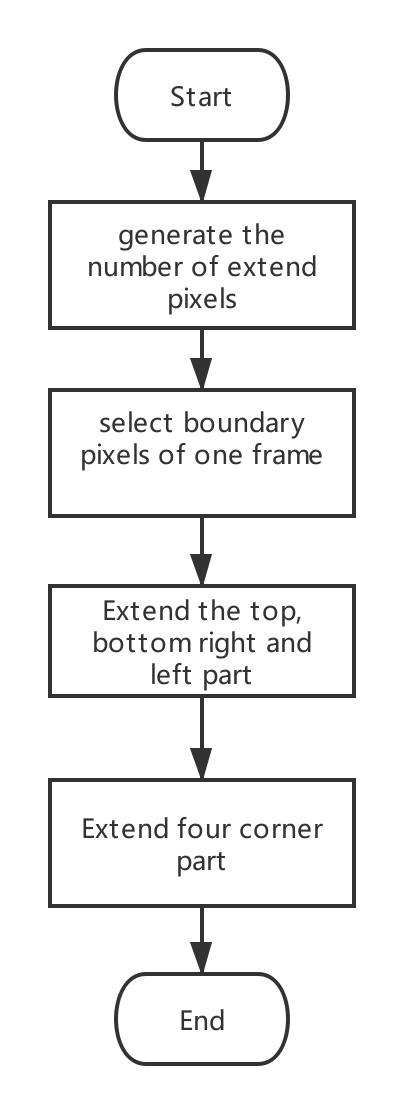


Figure 9: The whole process of zooming out a video of the dataset

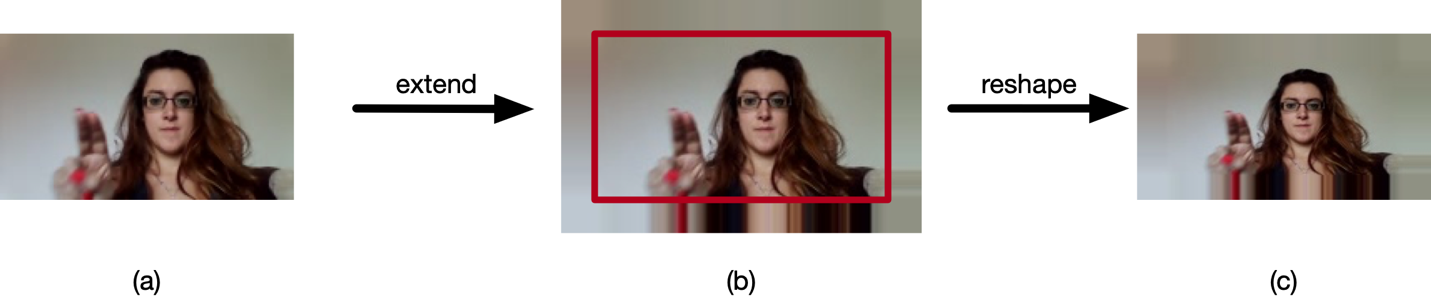


Figure 10: An example of zooming out a frame

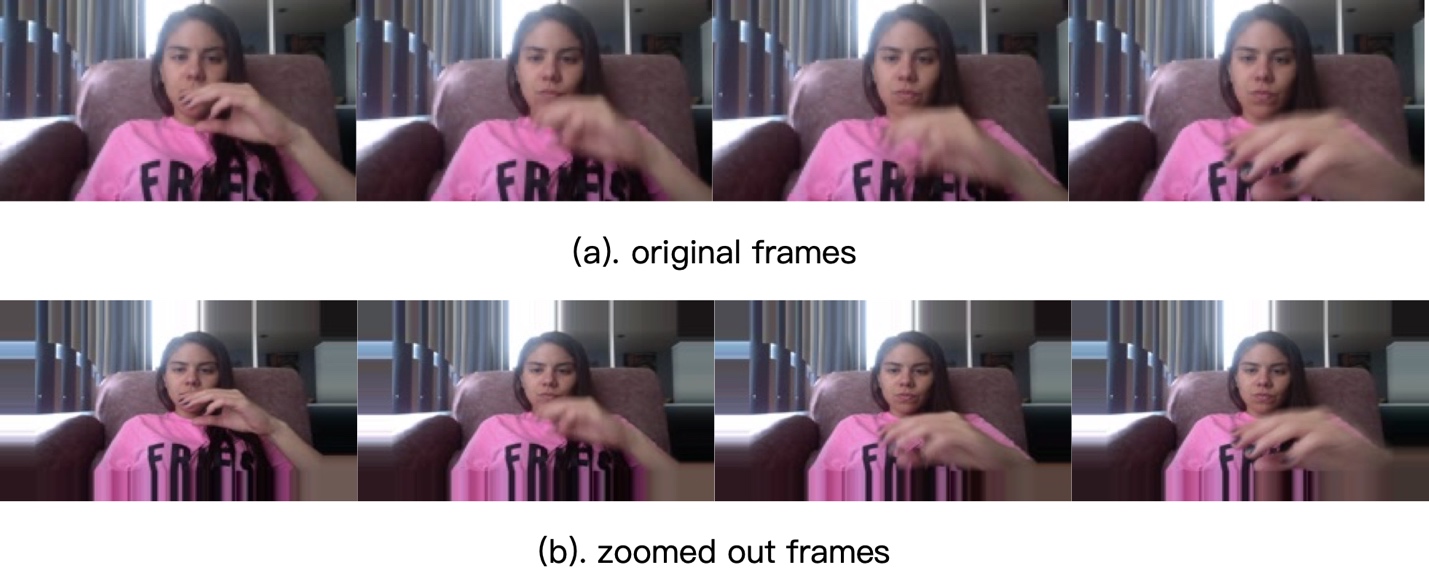


Figure 11: The comparison between the original dataset and the “” Jester dataset (a) is the original frame and (b) is the “zoomed-out” frame

## 4.3 Data preprocessing

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 12, 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 13.

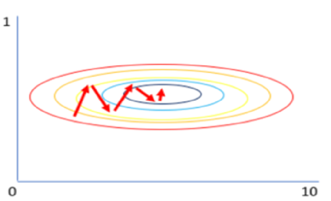


Figure 12: Gradient descent on different scale input

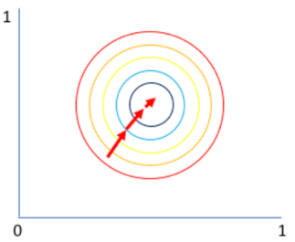


Figure 13: Gradient descent on the same scale input

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



, where  is the original input data,  is the mean of all the input data, and  is the standard deviation of it. In this study, we use this method to normalize the input data[30].

## 4.4 Training Details

This section introduces the experiment detail about hardware and software environment, and hyper-parameters about the learning phase.

### 4.4.1 Environments

We implemented our training and recognition system by using Python 3.5, PyTorch 1.0 and Cuda 10.0. To preprocess the video and images, we choose OpenCV 3 and pillow 5.3 library, because these previous libraries are friendly to Ubuntu Operation System, especially Cuda 10.0 and OpenCV 3.0.

The experiment hardware environment is as follows: Processor: Intel(R) Core(TM) i7-8700 3.20 GHz; System memory: 16GB (2400 MHz); GPU: NVIDIA Corporation GTX 1070 11GB.

### 4.4.2 Training configuration

We choose cross-entropy loss to calculate the model loss and the mini-batch stochastic gradient descent algorithm, mini-batch SGD, to optimize the parameters of the network. The batch size is 16 and momentum is 0.9. The dropout radio is set as 0.5. The network parameters are initialized with the pre-trained models, Inception v2, from ImageNet. The learning rate is set 0.001 in the beginning of the experiment and decrease to its when the loss fails to decrease compared with previous epoch training. The maximum optimization number of epochs is set as 50.

The number of groups may affect the model performance. Fewer groups could reduce the computation cost but it also reduces the accuracy of the model. On the contrary, more groups could increase the computation cost and meanwhile the accuracy of the model increase as well. Thus, we should find a balance between model accuracy and computation cost by choosing a suitable number of groups. For this purpose, we set the number of groups like 3, 5, 7, 9, 11, 13 and 15 and compare their performance.

## 4.5 Results and Analysis

### 4.5.1 Change of loss

The goal of training is to minimize the loss by using the gradient descent algorithm. During the training process, the parameter update depends on the cross-entropy loss. Thus, the change of loss is very useful to make sure that the model training is effective. As Figure 14 shows, the loss of the model has been on a downward trend, which indicates that the training process is effective.

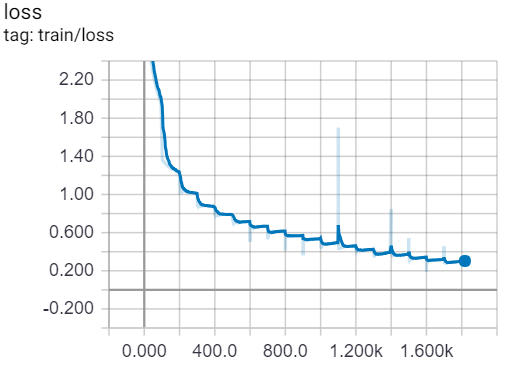


Figure 14: The change of loss

### 4.5.2 Accuracy performance

In order to find a suitable group number for our approach, we set the group number as 3, 5, 7, 9, 11, 13 and 15 to compare their accuracy. The result of the comparative experiment was shown in Figure 15. When the group number increase from 3 to 9, the accuracy of the model has been increasing but it doesn’t keep on increasing although the group number is increased. However, it cost much more computation time. According to this experiment, our model could get the best performance with an accuracy of 95.41%, when the group number is 9. The result on the “zoomed-out” Jester dataset is shown in Figure 16. Comparing with the result on Jester Dataset, we know that our approach achieves approximately the same accuracy when the group number is 9. However, the model trained by the “zoomed-out” Jester dataset should have better robustness and practicability.

Figure 15: Result on the validation set of Jester Dataset with the different group number. X-axis: number of groups Y-axis: accuracy

Figure 16: Result on the validation set of the “zoomed-out” Jester Dataset with different group numbers. X-axis: number of group, Y-axis: accuracy

### 4.5.3 Comparison with other approaches

Comparing with other approaches, our approach, STSNN, achieves a relatively good result with 95.41% accuracy. The Fusion\_TSN \_LSTM approach is in the first place as Table 3 shows. According to the name of the approach, this approach combines two neural network structures which means that this approach is more complex than our approach. Our approach could train on the environment with 16GB memory and a single GPU GTX 1070. Thus, our approach may be better than that in the area of computation cost. Comparing with 3D CNN architecture, this approach only need to train a CNN architecture which is pre-trained model and the fully connected layer which means that this number of parameters trained by this method is equivalent to the parameter number of fully connected layer of 3D CNN.

Table 3: Comparison with other approaches

|  |  |
| --- | --- |
| Approach | Accuracy (%) |
| Fusion\_TSN\_LSTM | 97.09 |
| RFEEN | 97.06 |
| MFFs | 96.28 |
| ResNext 101 | 95.87 |
| 3D CNN | 94.85 |
| STSNN | 95.41 |

## 4.6 System testing

During the testing phase, we choose a normal room as our testing background. The different colors of the background and the bright light on the right side were our testing challenges in Figure 17. After all the hand gestures were tested, the result indicated that the model could accurately recognize hand gestures in the complex contexts and the robustness of our model is proved.

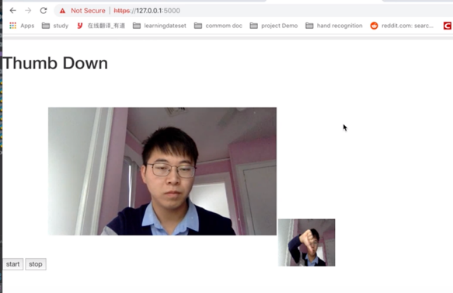


Figure 17: the testing screenshot

# Chapter 5 – Conclusion

This study proposed a new short-term sampling neural network for hand gesture recognition. We sample the frame and optical flow from each group and combine the features to predict the hand gesture. There are three distinguished advantages of this new method:

1. It reduced the input number of video frames by using group sampling
2. It reduced the training parameters by sharing CNN parameters.
3. It fixed the model input regardless of video length.

The developed system based on the new algorithm was trained and evaluated on the Jester dataset. To test robustness of the new approach, we zoomed out the Jester dataset by coping the boundary of original images. We achieved an average accuracy of 95.41%, 95.39% on the Jester dataset and the “zoomed-out” Jester dataset, respectively. This experiment result shows that the short-term sampling approach is effective for hand gesture recognition.

Future work of this study is as follows:

* We would like to apply our approach to some other challenging video analysis tasks. In this way, we can test our approach in other video analysis area.
* The optical flow approach plays a critical role in this study. However, the current optical flow algorithm cost too much computation. The deep learning methods could be used to calculate the optical flow picture.
* This study assumes that the video has already been trimmed and each video contains only one of hand gestures. This approach could apply to untrimmed video by setting a suitable group frames.

# Reference

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. W. Lu, Z. Tong and J. Chu, "Dynamic Hand Gesture Recognition With Leap Motion Controller," in IEEE Signal Processing Letters, vol. 23, no. 9, pp. 1188-1192, Sept. 2016.
5. Marin, G., Dominio, F. & Zanuttigh, P. Multimed Tools Appl (2016) 75: 14991. https://doi.org/10.1007/s11042-015-2451-6
6. G. Marin, F. Dominio and P. Zanuttigh, "Hand gesture recognition with leap motion and kinect devices," 2014 IEEE International Conference on Image Processing (ICIP), Paris, 2014, pp. 1565-1569.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. A. Ardakani, C. Condo, M. Ahmadi and W. J. Gross, "An Architecture to Accelerate Convolutional 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. 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
16. A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar and L. Fei-Fei, "Large-Scale Video Classification with Convolutional Neural Networks," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 2014, pp. 1725-1732.
17. K. Simonyan , A. Zisserman, Two-stream convolutional networks for action recognition in videos, Proceedings of the 27th International Conference on Neural Information Processing Systems, p.568-576, December 08-13, 2014, Montreal, Canada
18. Burton, Andrew; Radford, John (1978). Thinking in Perspective: Critical Essays in the Study of Thought Processes. Routledge. ISBN 978-0-416-85840-2.
19. Warren, David H.; Strelow, Edward R. (1985). Electronic Spatial Sensing for the Blind: Contributions from Perception. Springer. ISBN 978-90-247-2689-9.
20. Gibson, J.J. (1950). The Perception of the Visual World. Houghton Mifflin.
21. Royden, C. S.; Moore, K. D. (2012). "Use of speed cues in the detection of moving objects by moving observers". Vision Research. 59: 17–24. [doi](https://en.wikipedia.org/wiki/Digital_object_identifier):[10.1016/j.visres.2012.02.006](https://doi.org/10.1016%2Fj.visres.2012.02.006)
22. Zhang, G.; Chanson, H. (2018). "Application of Local Optical Flow Methods to High-Velocity Free-surface Flows: Validation and Application to Stepped Chutes" (PDF). Experimental Thermal and Fluid Science. 90: 186–199.
23. Farnebäck, Gunnar. (2003). Two-Frame Motion Estimation Based on Polynomial Expansion. In: Image analysis. 2749. 363-370. 10.1007/3-540-45103-X\_50.
24. W. Zhang and J. Wang, "Dynamic Hand Gesture Recognition Based on 3D Convolutional Neural Network Models," 2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC), Banff, AB, Canada, 2019, pp. 224-229.
25. L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: towards good practices for deep action recognition. In European Conference on Computer Vision, 2016. 8
26. S. J. Pan and Q. Yang, "A Survey on Transfer Learning," in IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, Oct. 2010.
27. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 2818-2826.
28. Srivastava, Nitish & Hinton, Geoffrey & Krizhevsky, Alex & Sutskever, Ilya & Salakhutdinov, Ruslan. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 15. 1929-1958.
29. J. Materzynska, G. Berger, I. Bax and R. Memisevic, “The Jester Dataset: A Large-Scale Video Dataset of Human Gestures”, 2019, ICCV
30. Grus, Joel (2015). Data Science from Scratch. Sebastopol, CA: O'Reilly. pp. 99, 100. [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [978-1-491-90142-7](https://en.wikipedia.org/wiki/Special:BookSources/978-1-491-90142-7).