CS6643 - Computer Vision - Final Project

Dongzi Qu (dq394), Zili Xie (zx979), Lynn Li (ml6589), Mina Lee (ml6543)

1 Introduction

Computer vision and visual recognition are some of the emerging technologies nowadays. Human-robot interactions have been getting more attention thanks to the recent artificial intelligence technologies. Hand gestures are a tool of communications, and understanding the gestures by computer could be applied in a variety of fields. However, it is easy for a computer to misunderstand the gesture due to the complex backgrounds, dynamic lighting, and different shapes of hands[2].

We aim to work on a project that not only converts images for applications but also integrates with the whole process of image processing. Hand gesture recognition system relates the overall process including real-time image processing, machine learning modeling and classification. This computer vision project describes a hand gesture recognition system based on video motion detection and deep learning recognition model.

2 Design

In this section, we provide the basic logic of workflow of our project. First of all, we decided a five gesture vocabulary; Fist, Rock, OK, Palm, and Victory.

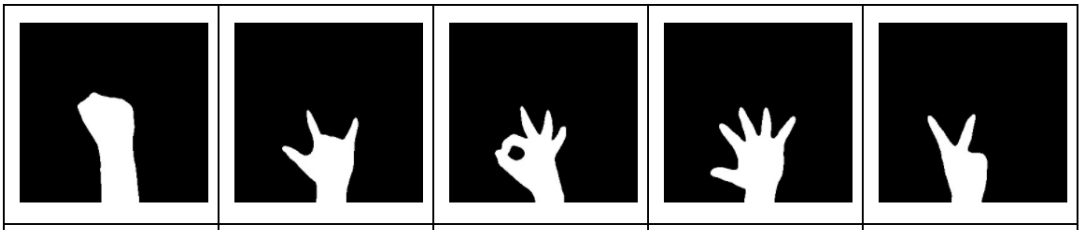

Figure 1: gesture samples

We use the camera on a laptop for image capturing. We have three steps to process the video streaming; captures the image, contour the image and extract the hand object. In order to have a clear hand gesture image, we created a box frame to locate the hand. The program reads the live streaming video and captures the object to remove the background. In order

to find out the edge of the object, Gaussian filters are used for smoothing and thresholding was applied for image detection. Then, the background is blacked out and only the hand object is captured with the white color, which can be seen from Figure 1.

After obtaining the white-black images for several kinds of gestures, contoured hand gesture image is applied to the deep learning model for recognition task. We trained CNN model with our own gesture images from local database, which will be discussed inside the data section. The sample images were gathered under different conditions, including using different sizes of hands, angles, and background. When the whole training process is over, we choose the best checkpoints and apply this model on the new input streaming gestures for prediction.

3 Methodology

In this section, we will talk about the specific methods about our implementation, including how to extract hands from video frame and how to build a CNN model for prediction.

3.1 Hand Extraction

Here are several useful steps for extracting the hand from video frame:

First, we create a box area on the frame as of our ROI (Region of Interest). This helps us to find out the location of the user's hand without considering the complex positions. After the program is executed, it takes 50 frames (2 to 3 seconds) to calculate the average background inside the ROI, which will be used in the next few steps.

Then, we mainly use *background subtraction*, *thresholding* and *contouring* to get the clear shape of hands. After we get the background value and put hands into ROI, program will measure the absolute difference between the background and the frame with hand. Then, the binary thresholding is applied on the subtraction given the difference of pixels where hand appears is larger than the other pixels. Through this process, we can get the white-black image of our ROI. Last but not the least, we draw the contour on it to adjust the shape of gesture shown as Figure 1.

In this step, we assume the computer is not moved and the background remains still accordingly. To avoid users moving their computers by accident, we also provide users with the functionality of background recalibration. Users don't need to re-compile the program in order to get a re-computed background. Also, this tool allows user to change the ideal background if the the current scene is not suitable for extracting hands.

3.2 Hand Recognition

After extracting hands from the video frame, we can move on to the prediction task. The issue of how to get the dataset for training and validating will be discussed in next section.

We will mainly talk about the model constructing in this section. Inspired from classification task on digits, we presumed the hand gesture recognition task shares lots of similarities with it, including same formatting in black-white image, the limited categories and so on. Therefore, we decide to use Convolution Neural Network (CNN) for prediction. Here, we list two different methods on how to build and utilize the CNN model:

Model 1: We build our first model by adapting from the VGG16; a talent CNN model from Keras, which can be used for image classification. As for VGG16, it is powerful on extracting features from input images and aggregating them later for prediction. However, the shape of its original output is (1000,1), supporting 1000 different labels classification task. Since we only have 5 categories, we add few more linear layers after the original model to transform the 1000 dimensions to 5.

After this stage, sampled gesture data is used as an input of the model and trained for several epochs. Notice that, we set the parameters from original VGG16 as non-trainable, which means we only need to train the parameters from the new added layers.

Some details: The shape of input gesture image should be resized to (224,224,3), which is the standard shape for VGG16. Moreover, one-hot encoding should be applied on the labels in order to eliminate the effectiveness from the absolute value of labels. More details about the inner framework of this model can be found inside our project.

Model 2: As for the second recognition model, we try not to use the pretrained model. Instead, we build our model following the similar ways as VGG16 but change the orders of layer and resize the output shape between some layers. VGG16 (Visual Geometry Group) is a CNN (Convolution Neural Net) architecture developed by University of Oxford.[4] It has 16 weighted layers. We chose this model because it is considered to be one of the most distinguished computer vision model architecture. VGG16 has 5 convolution layers group, each consists of 2 or 3 convolution layers. They focused on having convolution layers of 3x3 filter with a stride 1, always same padding, and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. For the output it has 3 FC (fully connected layers) followed by a softmax. This network is a pretty large network and it has approximately 134 million parameters. [handGestureVGGModel.ipynb]

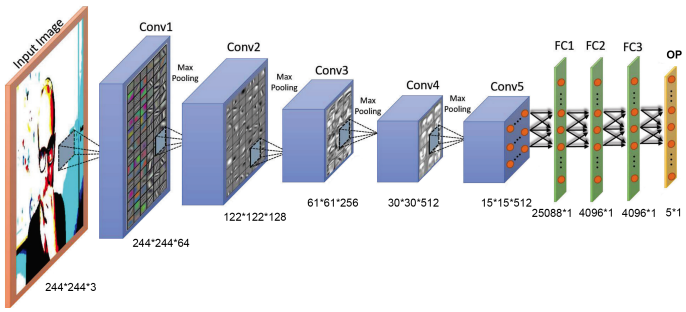

Figure 2: VGG16 Architecture [1]

4 Dataset

We obtained the dataset for training, validating and testing on our own. In the beginning of the project, we tried to use dataset from Kaggle's project[3]. However, the resolution of each sample from this dataset was large and the format was not well designed as black-white images. Due to the limitation in computing resources, we decide to build our own database rather than using this one.

The way of collecting sample images mainly followed the same path from section [3.1]. We were able to get the frame of gestures with a nice shape in black-white format. What we need to do is save all these frames into our local memory. Additionally, in order to make predicting model as robust as possible, we needed to increase the diversity of the gesture images. To achieve this goal, we gathered different images from 10 different people for each category. Therefore, we can assume that we have a variety of images in terms of size, shape or angles. In summary, our database contains 4000 images for all 5 categories, and all of them are in same size and format but in different looking. These images are divide into training set and validating set and then imported to the predicting model.[dataset]

5 Results

In this section, the results from two different models mentioned in [3.2] will be presented respectively.

Model 1: This model is adapted from VGG16, so the performance is mainly determined by the original model on classification task. As you can see from Figure 3, we set the number of epoch equals to 50 and the accuracy of model reaches the highest in the 6th epoch, which is more than 90 percent (validation accuracy reaches 91 percent). We utilize the checkpoints from this epoch because even though the training process does not reach convergence, the accuracy is getting decreased in 7th epoch. Additionally, this accuracy is very closed to the data from original VGG16 model.

```
Epoch 3/50
2500/2500 [==============================] - 1304s 522ms/step - loss: 0.5695 - accuracy: 0.7644 -
val_loss: 0.4677 - val_accuracy: 0.8580
Epoch 4/50
2500/2500 [==============================] - 1242s 497ms/step - loss: 0.4045 - accuracy: 0.8532 -
val_loss: 0.3280 - val_accuracy: 0.9100
Epoch 5/50
2500/2500 [==============================] - 1218s 487ms/step - loss: 0.3115 - accuracy: 0.8896 -
val_loss: 0.2637 - val_accuracy: 0.9160
Epoch 6/50
2500/2500 [==============================] - 1225s 490ms/step - loss: 0.2692 - accuracy: 0.9088 -
val_loss: 0.2564 - val_accuracy: 0.9140
Epoch 7/50
 384/2500 [===>..........................] - ETA: 14:29 - loss: 0.3173 - accuracy: 0.8776
```

Figure 3: Training process of Model 1

Here are some predicting results after applying this trained model on the video frames shown

as Figure 4. Once the hand is put into the ROI, the program draws the contour in blue lines for the hand region. Also, another small window shows the shape of gesture appears. Finally, you can see the prediction results from screen.

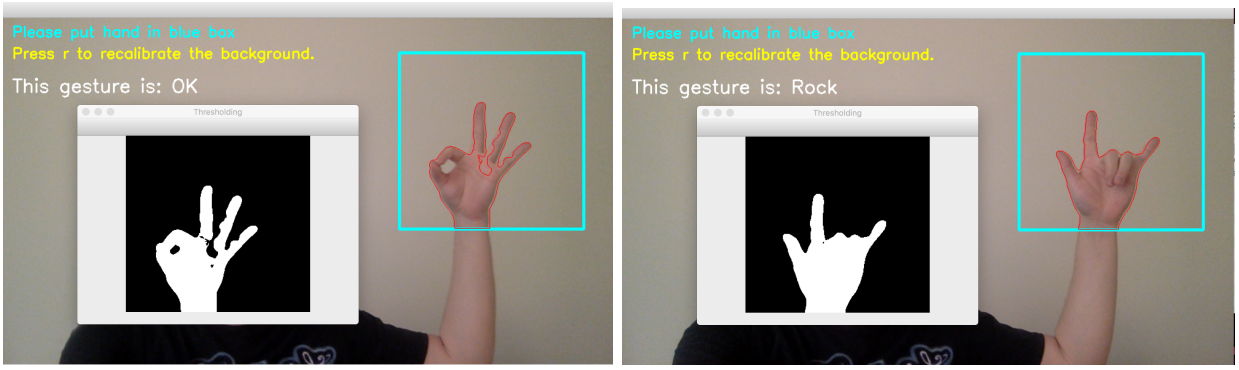

Figure 4: OK and Rock for Model 1

Model 2: Since the number of trainable parameters of this model is much larger than the model 1, it takes more epochs for training. We show the details of training process by means of line chart in Figure 6. Obviously, it takes more than 30 epochs for convergence. The final accuracy almost reaches 100 and 90 percent for the validation accuracy. As a whole, this model has better performance than the first one in terms of the accuracy. Our speculate for this high accuracy is that more parameters are considered in this model. Also, the model adapted from VGG16 might be too powerful for our task so the accuracy is a little bit lower than model 2. For more testing results, the video is uploaded in the following link. testing video (Figure 5)

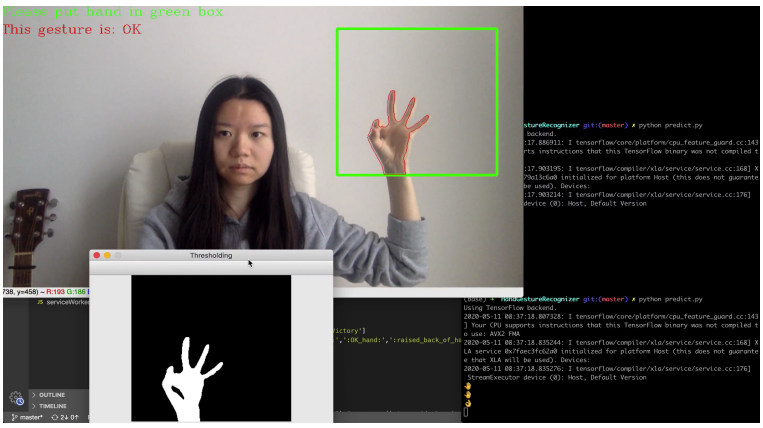

Figure 5: Testing scene of the second model

Here are some results corresponding to the second recognition model.

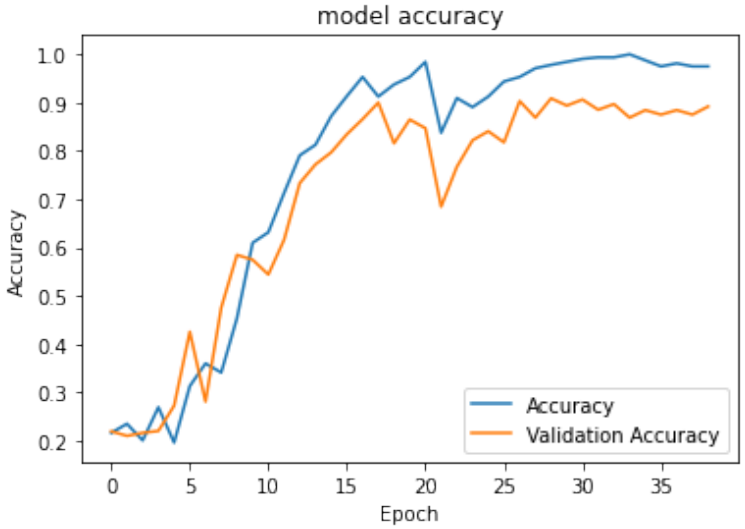

Figure 6: Training process for Model 2

Max Validation Accuracy: 0.91, Max Accuracy: 1, Min Loss: 0.006

6 Discussion

In this section, we are providing some analysis on our recognition model, including some challenging parts and the strengths/weaknesses of model itself.

6.1 Challenges

Firstly, it was challenging to capture the hands in a live streaming video as we cannot assume that the gesture is in the frame. In order to locate the hand gesture where we would read, we decided to have a box frame and ask the user to locate their hand in the box.

Then, capturing the hand from the focus box was also a challenging part. In the beginning, the program cannot get a clear black and white image depending on the background. If the background was not a clean white wall, the edge detection was not successful. We have tired tensorflow model for image detection and it performed well regardless of the background. However, the latency was high and the speed of detection was so slow that user would need to wait for 5 seconds to get a clear image. Instead, in order to improve the quality of extracting hands, we had to change the threshold for background subtraction or to change our background to more ideal one.

For the recognition part, the challenges were the parameter adjustment and the model construction. We experimented the model with different parameters to get the best result. The computing complexity was another obstacle when we trained each model. It takes about 2

hours (Macbook Pro, 8GB Memory) for the convergence of model, and repeating the training with different parameters added the time.

6.2 Strengths

Our gesture recognizer is robust in recognizing gesture input posing at different positions, angles and lighting. In the process of testing our model, we pose gestures at wide varying distances. In almost all the cases, our model gave us the right class of gestures in prediction. We believe that the method we use to collect the image can explain the robustness. When constructing the database of hand gesture images, we try to move and rotate our hand as much as possible to capture the shape of gestures from different view so that our model can learn how to recognize gesture in a more comprehensive way.

6.3 Weaknesses

Poor recognition accuracy under extreme light conditions. When the background is too bright, the contour of our hand gestures may become fuzzy and blurry due to the ambient light and lights reflected by other objects, making the light condition unstable. The threshold hand gesture images under this condition can be inaccurate, which directly leads to a low performance. That's why we create a function that allows recalibration. In addition, there can be slight latency when the model is making prediction. Last but not least, for now we can only recognize hand gestures inside of the ROI, instead of the full frame.

7 Conclusion and Future Work

In this project, we implement the hand gesture recognition task, which can automatically detect hand gestures from video frames and them recognize them to some specific labels. We can divide our project into two parts; detection and prediction. As for the detection, we mainly use the following tools: background subtraction, thresholding, contouring and filtering. Moreover, for the prediction, we build the recognition model using CNN in two different means, and both of them achieved the high accuracy.

In consideration of the potential future work, firstly, as we mentioned above the section [6.1], we may need to find another way for capturing the hand from ROI with better performance. If we dig more into the tensorflow model we mentioned in challenge section, we can expect better results with higher speed performance. In addition, as for the recognition task, we may be able to add more gestures and increase the size of our database. What's more, we can try some different models rather than CNN, like SVM or logistic regression for reducing the model training time.

References

[1] Asim Jan et al. "Artificial intelligent system for automatic depression level analysis through visual and vocal expressions". In: *IEEE Transactions on Cognitive and Developmental Systems* 10.3 (2017), pp. 668–680.

[2] Xingyan Li. "Gesture recognition based on fuzzy C-Means clustering algorithm". In: *Department of Computer Science. The University of Tennessee Knoxville* (2003).

[3] Leap Motion. *Hand Gesture Recognition Database*. URL: `https://www.kaggle.com/gti-upm/leapgestrecog`.

[4] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).