Hand Gesture Recognition

Aman Kumar (201911018) & Manan Gajjar (201911027)
Course: **Computer Vision - IT524**
By: Prof. Pankaj Kumar
Dhirubhai Ambani Institute of Information and Communication Technology

***Abstract*—Gesture recognition is a growing field of computer vision. As ways of interaction between humans and machines are widening, adequate solutions are in demand for better, easy, and fast communications. Since hand gestures are capable of carrying out enriched information, it is widely being used in robot control, intelligent household appliances, and other aspects. The paper realizes hand segmentation by topological analysis of binary image from opencv and with the help of background extracted from previous layers. Proposed deep learning model will then recognize the gesture from segmented region.**

***Index Terms*—computer vision, background subtraction, gesture recognition, deep learning, dataset generation.**

I. Introduction

Gesture recognition has been very interesting problem in computer vision community. It's accurate, stable, time saving and touch-less user interface has applications in home automation, gaming sector, sign language translation and many more.

Many different approaches has been proposed in the literature, each one can be classified into either 3D model based algorithms which uses skeletal models or appearance based models [1]. Appearance based models use image sequences as gesture templates, and then do gesture recognition using a sensor, shape tracing using geometric formulas, color-based methods, foreground segmentation with contour recognition or other methods.

II. Problem Statement

we are trying to recognizing these gestures from a live video feed. It is necessary to first extract the hand region and to remove all the unwanted portions from each frames. Later, this extracted feature need to be classified into corresponding class, for which we aim to use deep learning approach by predicting the gesture from trained deep learning model. Thus entire problem could be solved into two steps:

- find and segment hand region from video frames III-A,III-B
- predict the gesture class from trained neural network III-C

III. Experimentation

The live feed is captured by webcam using opencv's videoCapture and in a loop frames are extracted using camera.read functionality. Instead of recognizing gestures from all over the frame, we minimize the effective recognition region where algorithm has to look for hand. We highlight this region with red rectangle.

A. Foreground extraction

The first step is to separate foreground from background efficiently [2]. To do this we are using concept of running average. We make algorithm to capture a particular scene for 30 frames. During this time, we compute the running average [3] of the current frame and previous frame. formula for running average used is given as below

$$dst(x, y) = (1 - a) \cdot \mathrm{dst}(x, y) + a \cdot \mathrm{src}(x, y) \tag{1}$$

where, a is the weight of the source image, which is taken as 0.5. The final value of the $dst(x, y)$ after 30 frames is considered as the background.

After figuring background model using above technique, we get the current frame which contains the hand in addition to the background, and calculate the absolute difference between them. this difference image gives us the information about newly added foreground object in the frame.

B. Thresholding and Contour Extraction

Now to get exact region containing the hand, we threshold the difference image. This will also dampen other unwanted regions by assigning them value zero. The math behind thresholding is simple as

$$x(n) = \begin{cases} 1, & \text{if } n >= threshold \\ 0, & \text{if } n < threshold \end{cases} \tag{2}$$

If the pixel intensity of input is less than threshold, which was taken as 25 here, we will set it to value 255 or else 0.

Next step is to find contours in the thresholded difference image, which is achieved by opencv's findContours [4] function. We select the contour with the largest area which is assumed to be our hand.

C. Gesture Recognition

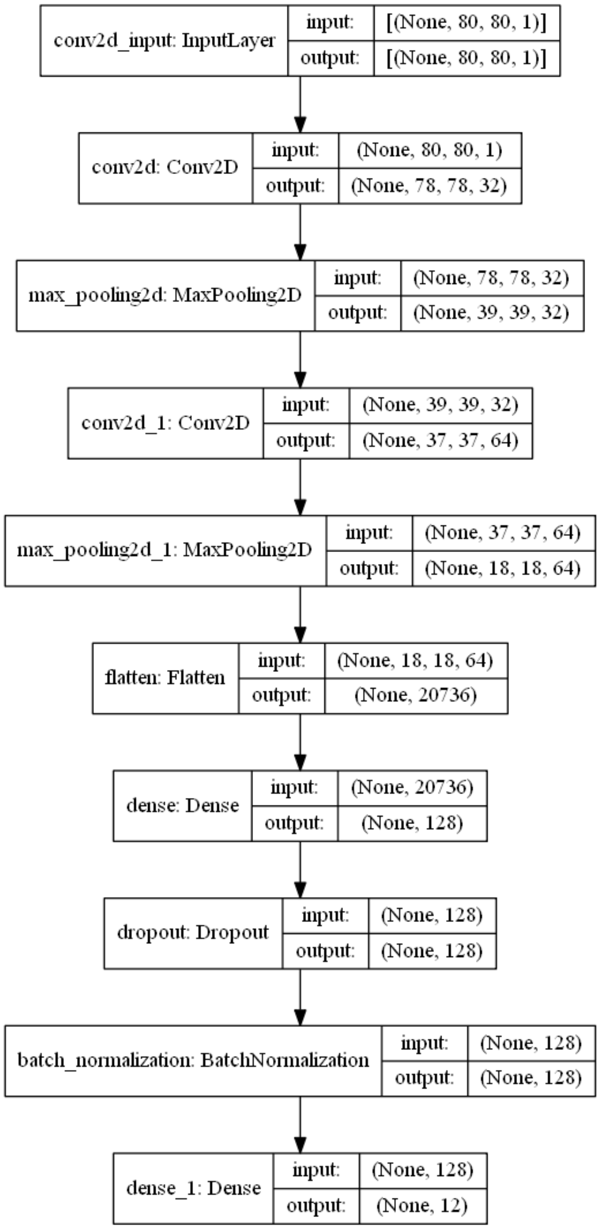
The segmented image is passed into a trained convolutional neural network. Since image is already binary we don't need to include segmentation subnet. The configuration of the network is as following.

- Network consists of initial two convolution layers with 32 and 64 number of neurons with ReLU activation respectively.
- Each convolution layer if followed by max-pooling for reducing spatial size and dealing with over-fitting as well.
- Next layer creates a dense layer with the flattened output and dropout and batch-normalization is applied.

- Last output layer uses softmax activation function to classify the image.

Model is compiled with adam optimizer and binary-cross entropy loss and accuracy metrics are applied for evaluation [5].

We are using custom generated dataset which is discussed in the section IV. Network is trained across 10 epochs with batch size of 32, as there is no increase in accuracy after that. Model was trained with loss of 0.02 and accuracy of 0.98.



conv2d_input: InputLayer | input: [(None, 80, 80, 1)] | output: [(None, 80, 80, 1)]

conv2d: Conv2D | input: (None, 80, 80, 1) | output: (None, 78, 78, 32)

max_pooling2d: MaxPooling2D | input: (None, 78, 78, 32) | output: (None, 39, 39, 32)

conv2d_1: Conv2D | input: (None, 39, 39, 32) | output: (None, 37, 37, 64)

max_pooling2d_1: MaxPooling2D | input: (None, 37, 37, 64) | output: (None, 18, 18, 64)

flatten: Flatten | input: (None, 18, 18, 64) | output: (None, 20736)

dense: Dense | input: (None, 20736) | output: (None, 128)

dropout: Dropout | input: (None, 128) | output: (None, 128)

batch_normalization: BatchNormalization | input: (None, 128) | output: (None, 128)

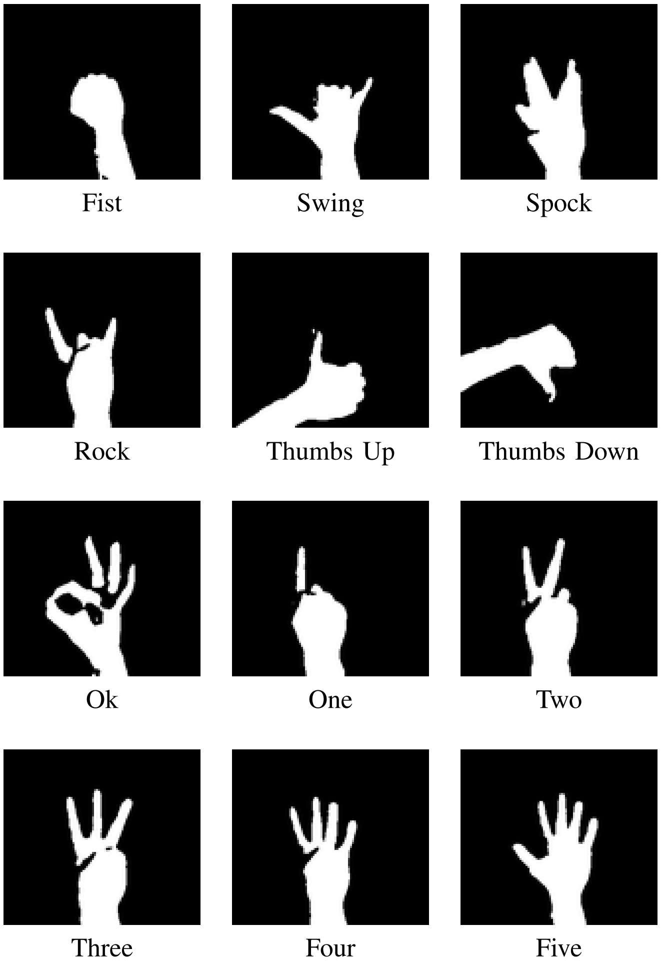
dense_1: Dense | input: (None, 128) | output: (None, 12)

Fig. 1. Network structure

IV. Dataset

We are generating a custom dataset by capturing images of different hand gestures. We have defined 12 different gestures for this experiment.

First, we are capturing those images by background elimination, as discussed in III-A, III-B. Algorithm will capture predefined number of images from the highlighted green rectangular region, resizes it, applies background elimination, and then save it in dataset under given label. We are resizing the images to (89×100) dimension. Our Dataset contains 1000 images of training for each class and 100 images of testing for each class. Samples from dataset are shown in Fig. IV. The ratio of training set to validation set is 10:1.



Fist

Swing

Spock

Rock

Thumbs Up

Thumbs Down

Ok

One

Two

Three

Four

Five

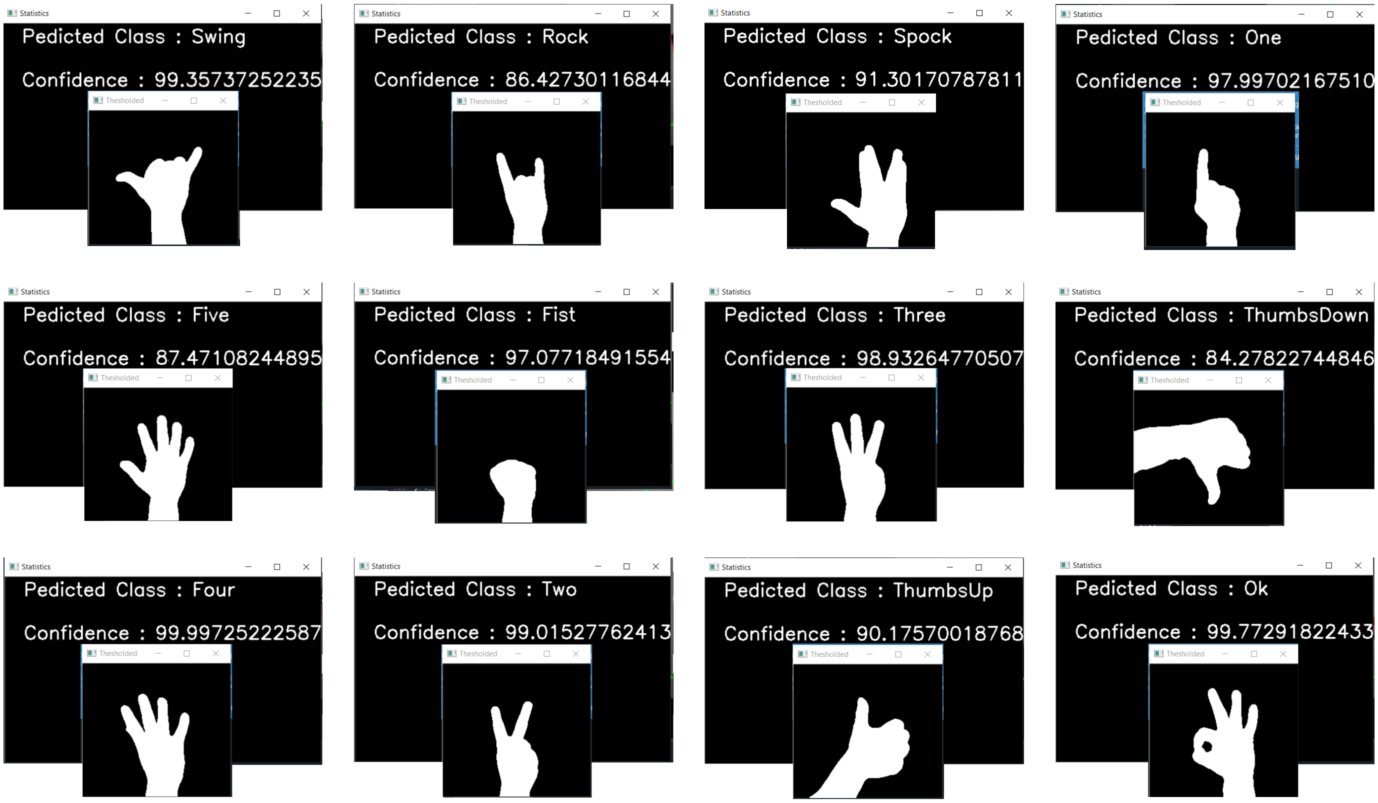
Fig. 2. Different Gestures

V. Results

The validation accuracy and validation loss for testing images from dataset are shown in Fig. V. The model achieves 0.82 accuracy and 1.22 loss on validation dataset. Some samples from prediction with confidence value are shown in Fig. V.

VI. Conclusion

Experiment shows that proposed model gives good result with both hand region recognition and gesture type detection. Network is giving around an average accuracy of 94% on different sets of gestures. Segmentation network like U-Net is not used here, because input image into the network is already



```
Pedicted Class : Swing
Confidence : 99.35737252235

Pedicted Class : Rock
Confidence : 86.42730116844

Pedicted Class : Spock
Confidence : 91.30170787811

Pedicted Class : One
Confidence : 97.99702167510

Pedicted Class : Five
Confidence : 87.47108244895

Pedicted Class : Fist
Confidence : 97.07718491554

Pedicted Class : Three
Confidence : 98.93264770507

Pedicted Class : ThumbsDown
Confidence : 84.27822744846

Pedicted Class : Four
Confidence : 99.99725222587

Pedicted Class : Two
Confidence : 99.01527762413

Pedicted Class : ThumbsUp
Confidence : 90.17570018768

Pedicted Class : Ok
Confidence : 99.77291822433
```

Fig. 3. Gestures predictions with confidence values

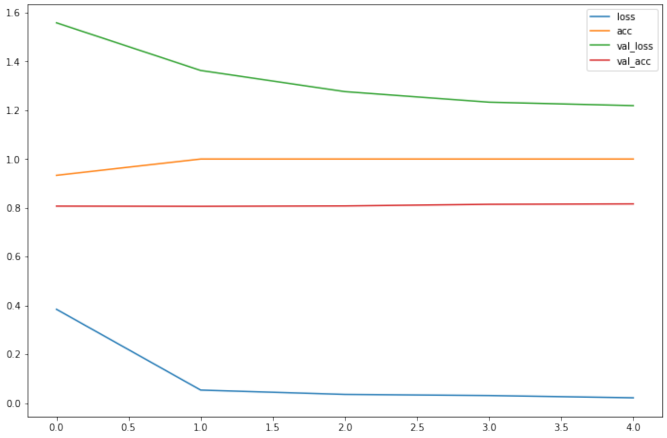


Fig. 4. Model performance graph

segmented. Although, instead of marking foreground as white, original RBG information can be used from foreground extraction to train a deep learning model and learn coarse features.

References

[1] Wikipedia, "Gesture recognition." [Online]. Available: https://en.wikipedia.org/wiki/Gesture_recognition

[2] opencv, "How to use background subtraction methods," Documentations. [Online]. Available: https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html

[3] ——, "Motion analysis and object tracking," Documentations. [Online]. Available: https://docs.opencv.org/3.0-beta/modules/imgproc/doc/motion_analysis_and_object_tracking.html

[4] ——, "Structural analysis and shape descriptors," Documentations. [Online]. Available: https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html

[5] tensorflow, "Tensorflow core v2.4.0," Documentations. [Online]. Available: https://www.tensorflow.org/api$_docs/python/tf$