Real-time AMERICAN SIGN LANGUAGE Recognition with CNN

[Arjun Singh Kanwal (MT201726)]

November 24, 2018

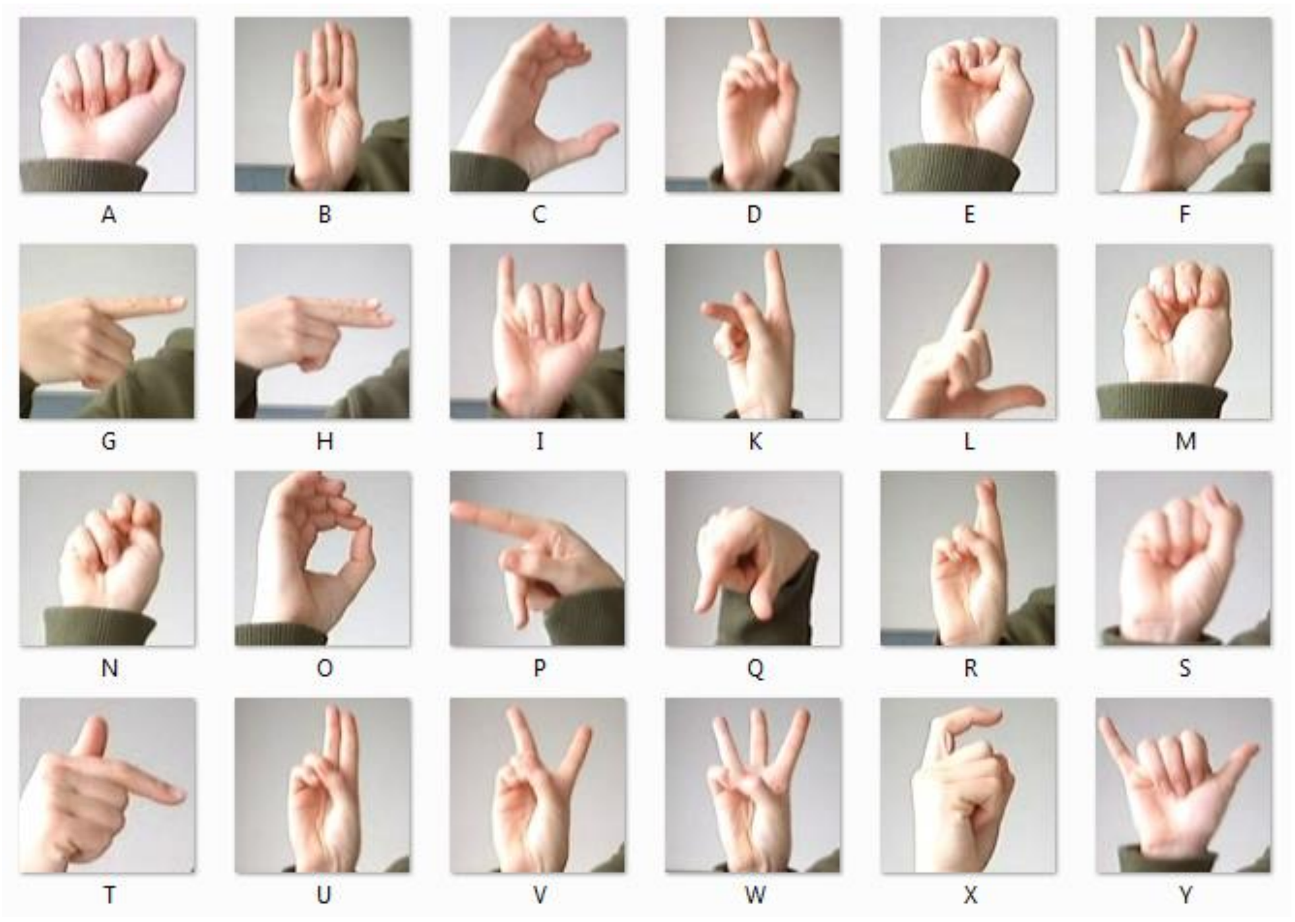
Contents

1. Introduction

Very few people understand sign language. Moreover, contrary to popular belief, it is not an international language. Obviously, this further complicates communication between the Deaf community and the hearing majority. The alternative of written communication is cumbersome, because the Deaf community is generally less skilled in writing a spoken language. Furthermore, this type of communication is impersonal and slow in face-to-face conversations. For example, when an accident occurs, it is often necessary to communicate quickly with the emergency physician where written communication is not always possible.

In order to diminish this obstacle and to enable dynamic communication, we present an ASL recognition system that uses Convolutional Neural Networks (CNN) in real time to translate a video of a user's ASL signs into text. American Sign Language (ASL) substantially facilitates communication in the deaf community.

The purpose of this work is to contribute to the field of automatic sign language recognition. We focus on the recognition of the signs or gestures.



Our problem consists of three tasks to be done in real time:

1. Obtaining video of the user signing (input).

2. Classifying each frame in the video to a letter.

3. Reconstructing and displaying the most likely word from classification scores (output).

2. PROPOSED METHOD

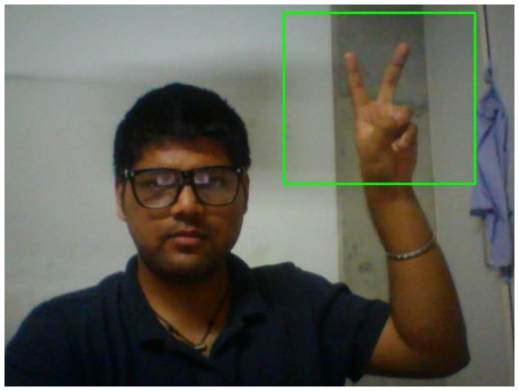
2.1. Dataset

I use the dataset by capturing data from web cam of my own laptop. More specifically capturing only the portion of image where my hand is present. My dataset consists 24 different gestures performed by me and one of my friends. I used 624 gestures 26 for each class. I have 24 total classes.

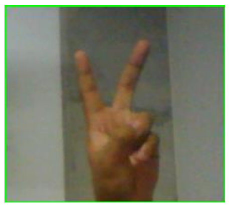
2.1. Preprocessing (Algorithm = Image Subtraction)

Our first step in the preprocessing stage is creating a box in frame of camera where user can put hand to give as input. This portion of image is first captured without any hand. Now when user put the hand in this portion, this image is subtracted with original image. The steps are as follows: -

Step-1. Capture the image using webcam.



Step-2. Extract hand portion.



Step-3. Subtract this image with background image.

Step-4. Convert the subtracted image into black and white.

Step-5. Apply GaussianBlur in above image.

Step-6. Apply threshold value such that all the pixel value less then 10 are converted to 0(black) and rest pixel value are converted to 255(white). The image is as shown.

These images are then feed to CNN.

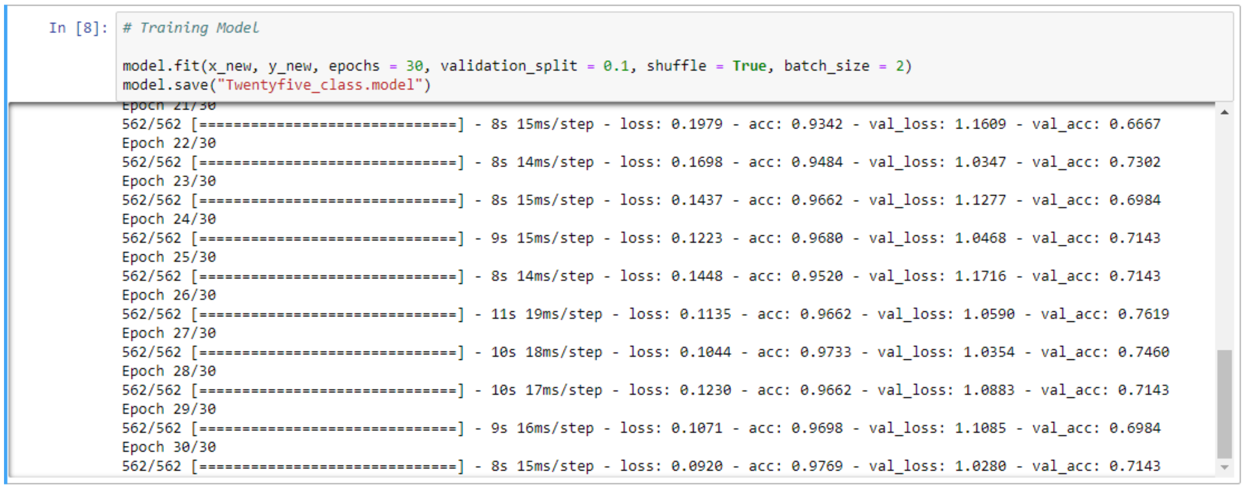
3. Convolutional Neural Network:

I have used keras for creating CNN. Keras is an abstraction library that can use tensorflow(by Google), Theano(by MILA Lab) or CNTK(by Microsoft). Because of TensorFlow popularity keras by default uses TensorFlow in backend.

3.1. Layers:

The architecture of model consists of one CNN. This CNN has 3 layers of convolution and 1 dense layer and finally SoftMax layer(output layer). I have used categorical crossentropy loss function and SGD optimizer (alpha = 0.0001).

3.2. Training:



```
In [8]: # Training Model

        model.fit(x_new, y_new, epochs = 30, validation_split = 0.1, shuffle = True, batch_size = 2)
        model.save("Twentyfive_class.model")
```

```
Epoch 21/30
562/562 [==============================] - 8s 15ms/step - loss: 0.1979 - acc: 0.9342 - val_loss: 1.1609 - val_acc: 0.6667
Epoch 22/30
562/562 [==============================] - 8s 14ms/step - loss: 0.1698 - acc: 0.9484 - val_loss: 1.0347 - val_acc: 0.7302
Epoch 23/30
562/562 [==============================] - 8s 15ms/step - loss: 0.1437 - acc: 0.9662 - val_loss: 1.1277 - val_acc: 0.6984
Epoch 24/30
562/562 [==============================] - 9s 15ms/step - loss: 0.1223 - acc: 0.9680 - val_loss: 1.0468 - val_acc: 0.7143
Epoch 25/30
562/562 [==============================] - 8s 14ms/step - loss: 0.1448 - acc: 0.9520 - val_loss: 1.1716 - val_acc: 0.7143
Epoch 26/30
562/562 [==============================] - 11s 19ms/step - loss: 0.1135 - acc: 0.9662 - val_loss: 1.0590 - val_acc: 0.7619
Epoch 27/30
562/562 [==============================] - 10s 18ms/step - loss: 0.1044 - acc: 0.9733 - val_loss: 1.0354 - val_acc: 0.7460
Epoch 28/30
562/562 [==============================] - 10s 17ms/step - loss: 0.1230 - acc: 0.9662 - val_loss: 1.0883 - val_acc: 0.7143
Epoch 29/30
562/562 [==============================] - 9s 16ms/step - loss: 0.1071 - acc: 0.9698 - val_loss: 1.1085 - val_acc: 0.6984
Epoch 30/30
562/562 [==============================] - 8s 15ms/step - loss: 0.0920 - acc: 0.9769 - val_loss: 1.0280 - val_acc: 0.7143
```

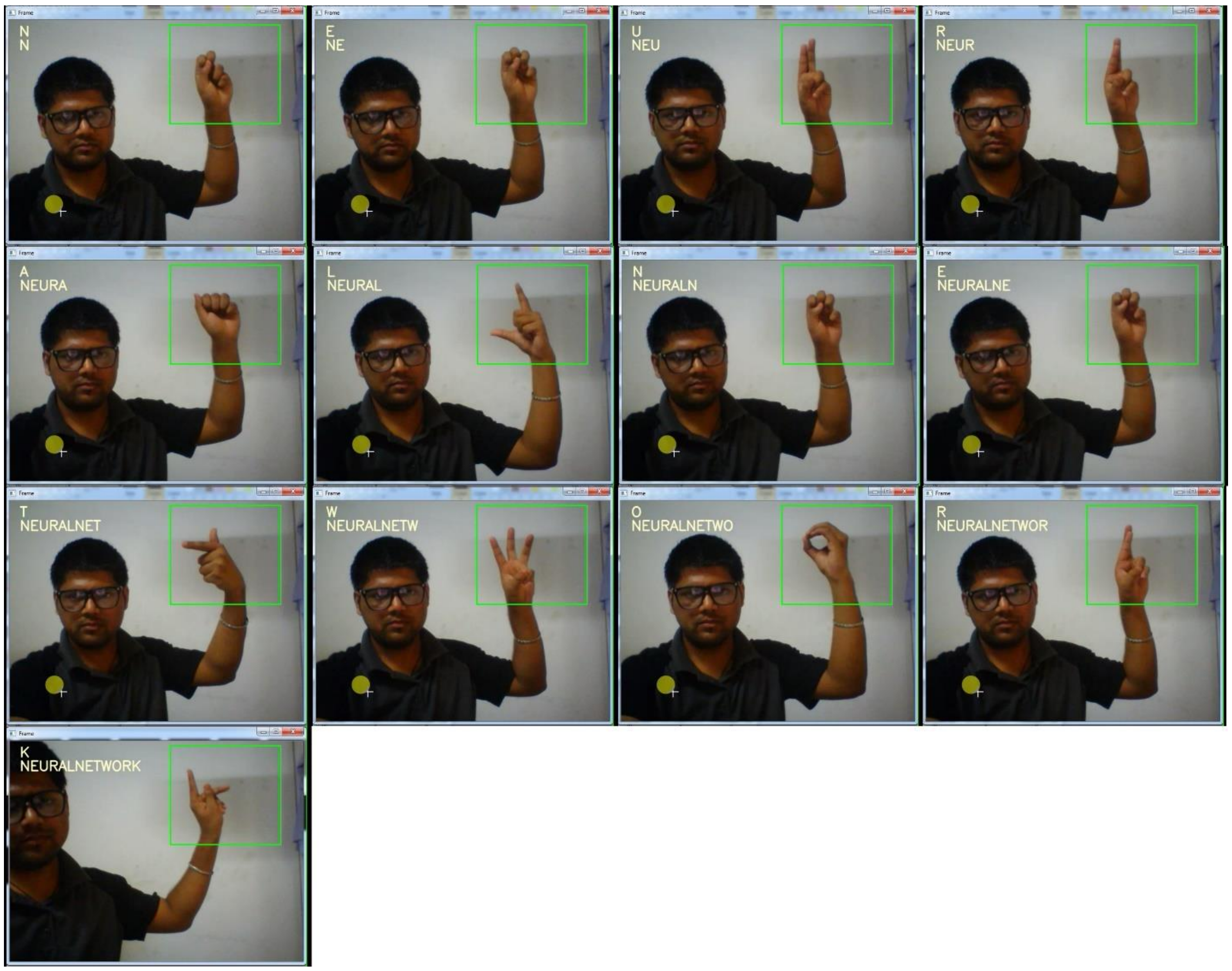
The validation results of these experiments are shown in the above image(last column). The training accuracy is around 96% and validation accuracy is 71%.

3.3. Testing:

For testing purpose, we are directly using live feed from webcam. The video is consisting of frames. We are passing each frame to our neural network and predicting the labelled output.

4. Result:

The top left denotes what letters the current image predicts and below it the words formed by the letters. The following demo gives the illustration of how to write keyword ‘Neural Network’.



5. Conclusion:

This work shows that convolutional neural networks can be used to accurately recognize different signs of a sign language, with users and surroundings not included in the training set. This generalization capacity of CNNs in spatiotemporal data can contribute to the broader research field on automatic sign language recognition.

6. Project links

GitHub link: https://github.com/Arjun8900/Neural_Network
YouTube Link: https://youtu.be/NBzqY9tJd7M