INDIVIDUAL FINAL REPORT

Introduction

Hand gesture is one of the methods used in sign language for non-verbal communication. It is most commonly used by people with speech and hearing disabilities for communication in their daily conversation activities. In this project we used ResNet for best metrics and for accurate results.

I initially started working on Keras to save best model and ended up with testing accuracy of 91% and model predicts the images well but, in recognizing some gestures for letters (U & V), (N & O) it failed to give accurately.

Background Information

When we are formatting images to be inputted to a Keras model, we must specify the input dimensions. Many datasets like MNIST are all conveniently the same size, of (32 x 32) or (28 x 28). Convolutional Neural Networks with Keras framework is used for image recognition.

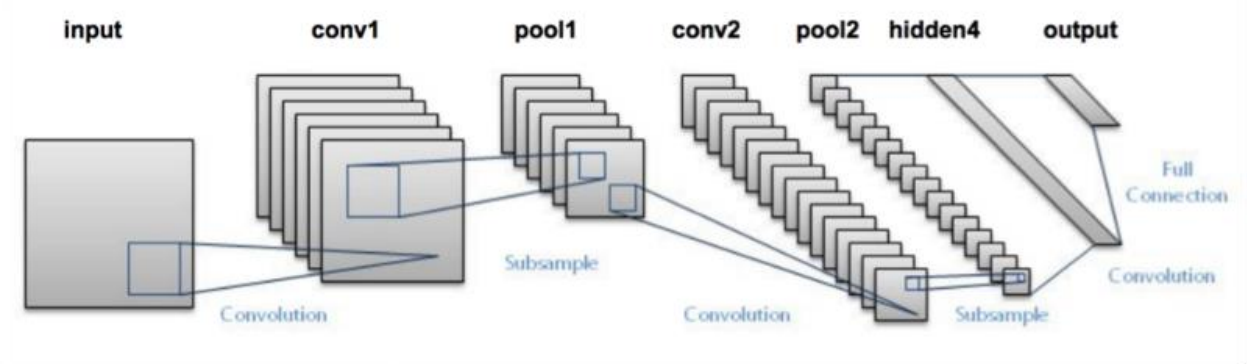


Fig: 1

CNN is a sequence of layers and each layer is used to transform one part of activation to another through some differentiable function f(x). For example, if we have (nxn) matrix and multiply with matrix dimension of (kxk), the matrix with dimension would be as below with stride(s)

$$\frac{(n-k+1)}{s} x \frac{(n-k+1)}{s}$$

Portion of work in project

I did work with Keras framework by building model using convolutional neural network. In preprocessing section train data is converted to numpy array, expanded its dimensions so that

model gets trained well and finally reshaped images. For converting multi-class labels to binary labels, LabelBinarizer( ) is used which makes the process easy with transform method.

Worked on group report and Power point presentation.

Results

After preprocessing like converting train data to numpy array, expanding image dimensions and reshaping, splitted train data into x_train, y_train, x_test, y_test with random state as 40.

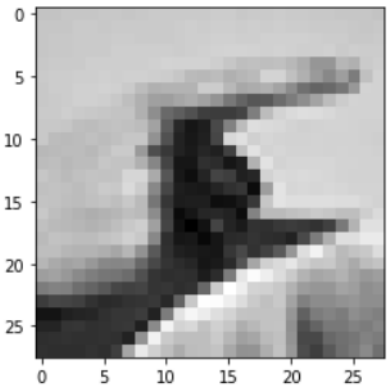
For building model CNN is used with 3 convolutional layers, flatten layer and 2 dense layers are used for saving best model.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 64)        640
_________________________________________________________________
dropout_1 (Dropout)          (None, 26, 26, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 64)        36928
_________________________________________________________________
dropout_2 (Dropout)          (None, 11, 11, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
dropout_3 (Dropout)          (None, 3, 3, 64)          0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_2 (Dense)              (None, 24)                3096
=================================================================
Total params: 85,912
Trainable params: 85,912
Non-trainable params: 0
_________________________________________________________________
```

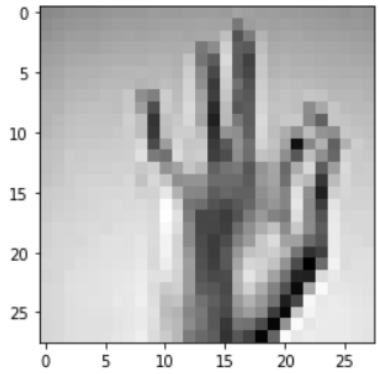
Flatten layer is used to have the shape that is equal to number of elements that are present in array without inclusive of batch dimension which is then passed to Dense Layer.
Dense layers is used to add the fully connected layer to the neural network.
Flatten-64, dense_1 – 128 (64*128+128) which gives 8320, dense_2-24 (128*24+24) which gives 3096. Total params-85912 (640+36928+36928+8320+3096)

Learning rate with 0.003, optimizer as Adam, considering loss – categorical cross entropy and taking epochs = 5 model is build and saved as ‘CNN.hdf5’. When model ran against test data accuracy is 91% and predicted most of the images pretty well but, when predicted letters J, O and V it went wrong.

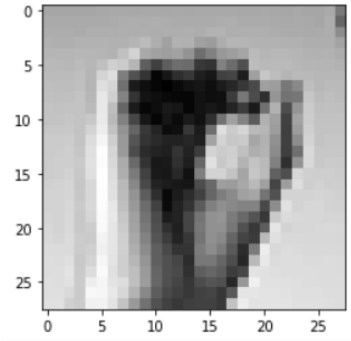
class: G

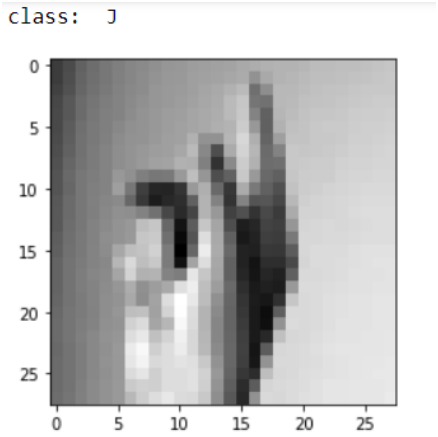


class: F



class: N





From above images we can see it predicted letter P as J and letter O as N. This shows there is some confusion in recognizing few letters which may be avoided by tuning hyper parameters.

Summary and Conclusion

The fine-tuned models clearly outperform the custom models. The failure in predicting few images can be of insufficient data to learn from or the learning happens to be very slow in the early epochs.

Percentage of code from Internet

Taken reference from class tutorials.

References

1. Ashok K Sahoo, Gouri Sankar Mishra and Kiran Kumar Ravulakollu. “Sign Language Recognition: State of the Art”, ARPN Journal of Engineering and Applied Sciences.
2. Ashish S. Nikam, Aarti G. Ambekar, “Sign language recognition using image-based hand gesture recognition techniques”.