English Alphabet classification from hand signs using CNN

COM511 – Pattern Recognition

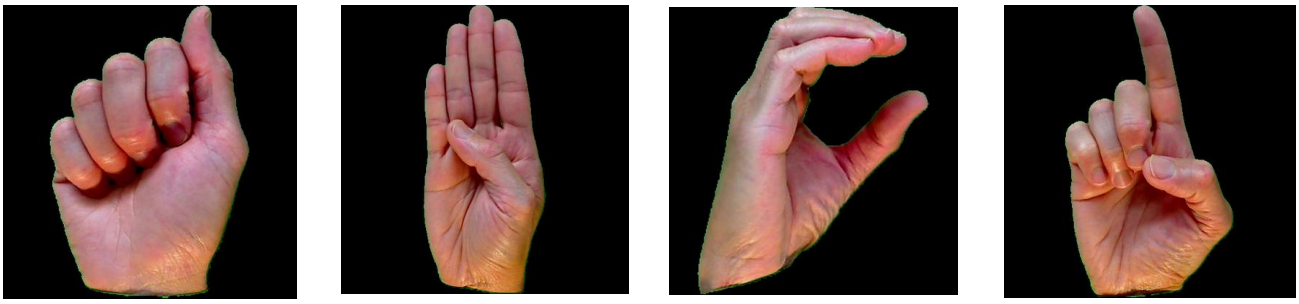
Submitted by,
Group 20
COE19D004 – Debarati Bhattacharjee

In this project 26 hand signs are used to classify 26 English alphabets with a CNN based model.

1. The Dataset

American Sign Language [https://www.kaggle.com/ayuraj/asl-dataset] dataset is used for classfication task. Only relevant folders corresponding to 26 English alphabets are used for this project. For each alphabet there are 70 samples except the letter 't' for which 65 samples are available.

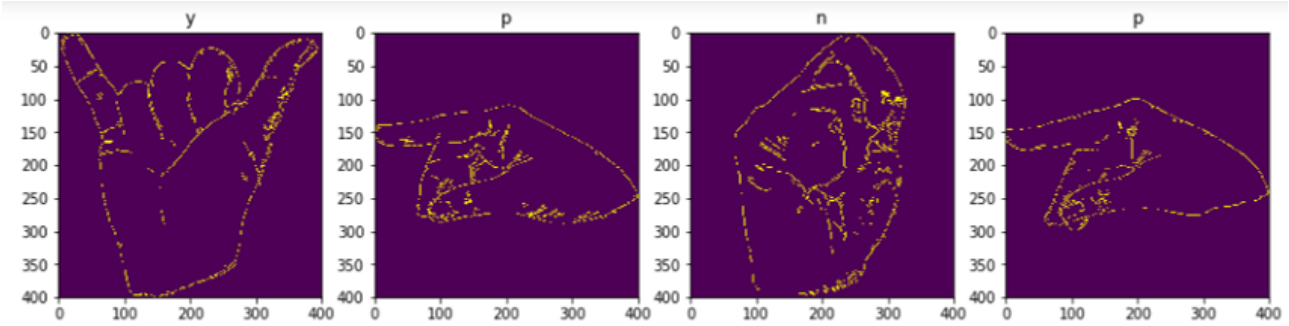
Sample images in dataset with their corresponding labels :



a b c d

2. Preprocessing

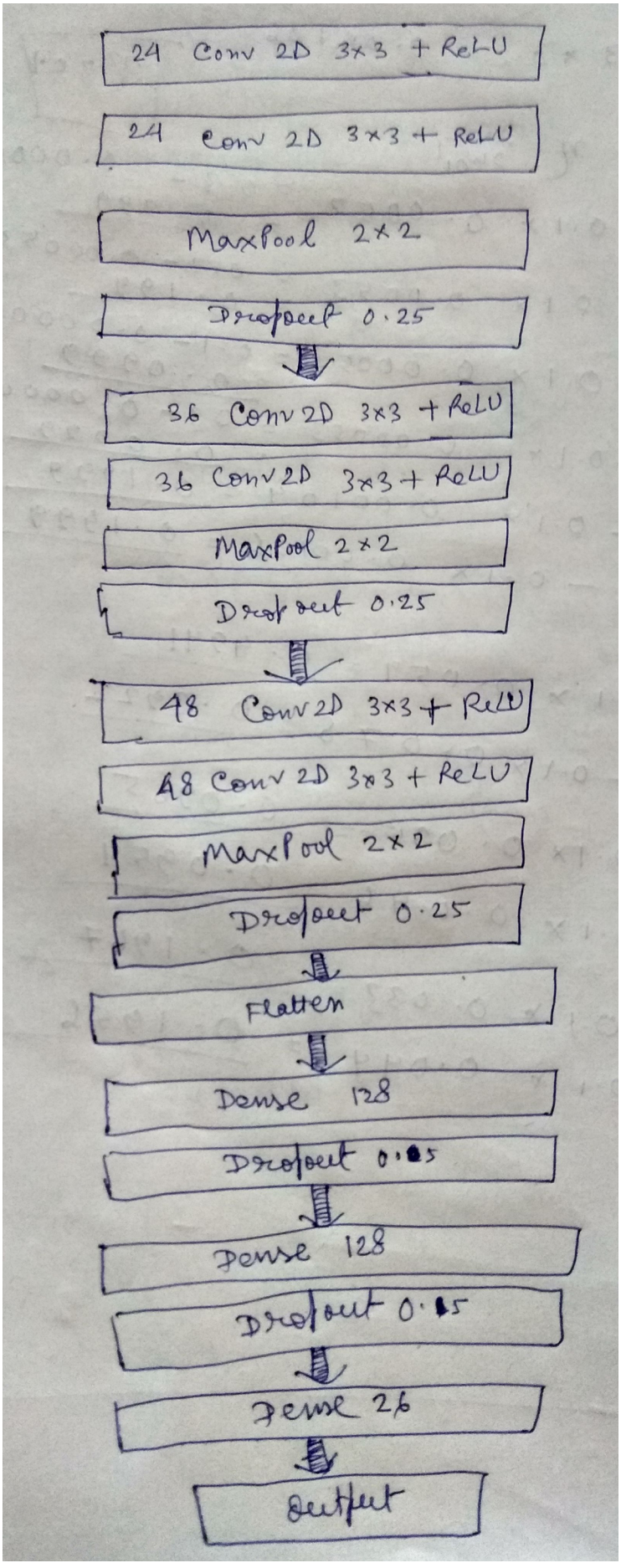
All images in dataset went through the Canny edge detector. The resultant images with detected edges are used to train, validate and test the classifier. Here are screenshots of few preprocessed images with their corresponding labels:



3. Data augmentation

Several data auudmentation techniques like rotation, width shift, height shift, shear, horizontal flip are used to increase the size and diversity of the dataset and avoid overfitting. The **ImageDataGenerator** class in **Keras** is used for this purpose.

4. The CNN architecture



API used : Keras Functional API
Layers : Input, Conv2D, MaxPooling2D, Dropout, Flatten, Dense
Activation functions : ReLU for innear layers, Softmax for output layer
Loss function : Categorical crossentropy
Optimizer : Adam
Kernel initializer : He Normal

5. The performance of classification model and output

5.1 : Training with early stopping

The classification model was trained with some specified early stopping criteria, maximum 350 epochs were allowed. The model got trained for 178 epochs and stopped due early stopiping acquiring 96.69% accuracy on test set.

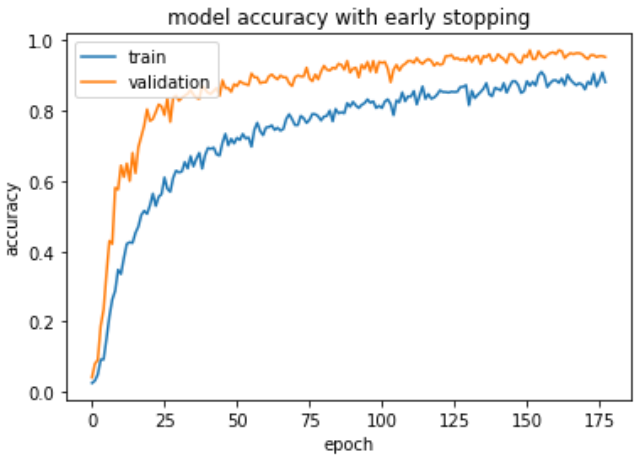
Early stopping code:

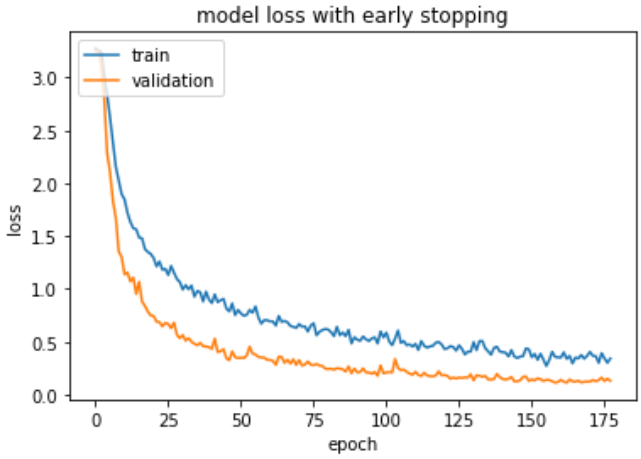
```
# early stopping
es = EarlyStopping(monitor='val_loss', min_delta=0.0001, mode='min', verbose=1, patience=15)
```

The last 5 epochs of training log:

```
Epoch 174/350
68/68 [==============================] - 64s 943ms/step - loss: 0.2976 - acc: 0.9072 - val_loss: 0.1372 - val_ac
c: 0.9587
Epoch 175/350
68/68 [==============================] - 63s 929ms/step - loss: 0.4337 - acc: 0.8577 - val_loss: 0.1611 - val_ac
c: 0.9532
Epoch 176/350
68/68 [==============================] - 63s 928ms/step - loss: 0.3409 - acc: 0.8878 - val_loss: 0.1288 - val_ac
c: 0.9559
Epoch 177/350
68/68 [==============================] - 63s 929ms/step - loss: 0.2968 - acc: 0.9108 - val_loss: 0.1510 - val_ac
c: 0.9559
Epoch 178/350
68/68 [==============================] - 63s 922ms/step - loss: 0.3437 - acc: 0.8715 - val_loss: 0.1305 - val_ac
c: 0.9532
Epoch 00178: early stopping
dict_keys(['val_acc', 'loss', 'val_loss', 'acc'])
```

Relevant curves:





Accuracy on test set:

Predicted class probabilities and labels(0 for 'a', 25 for 'z'):

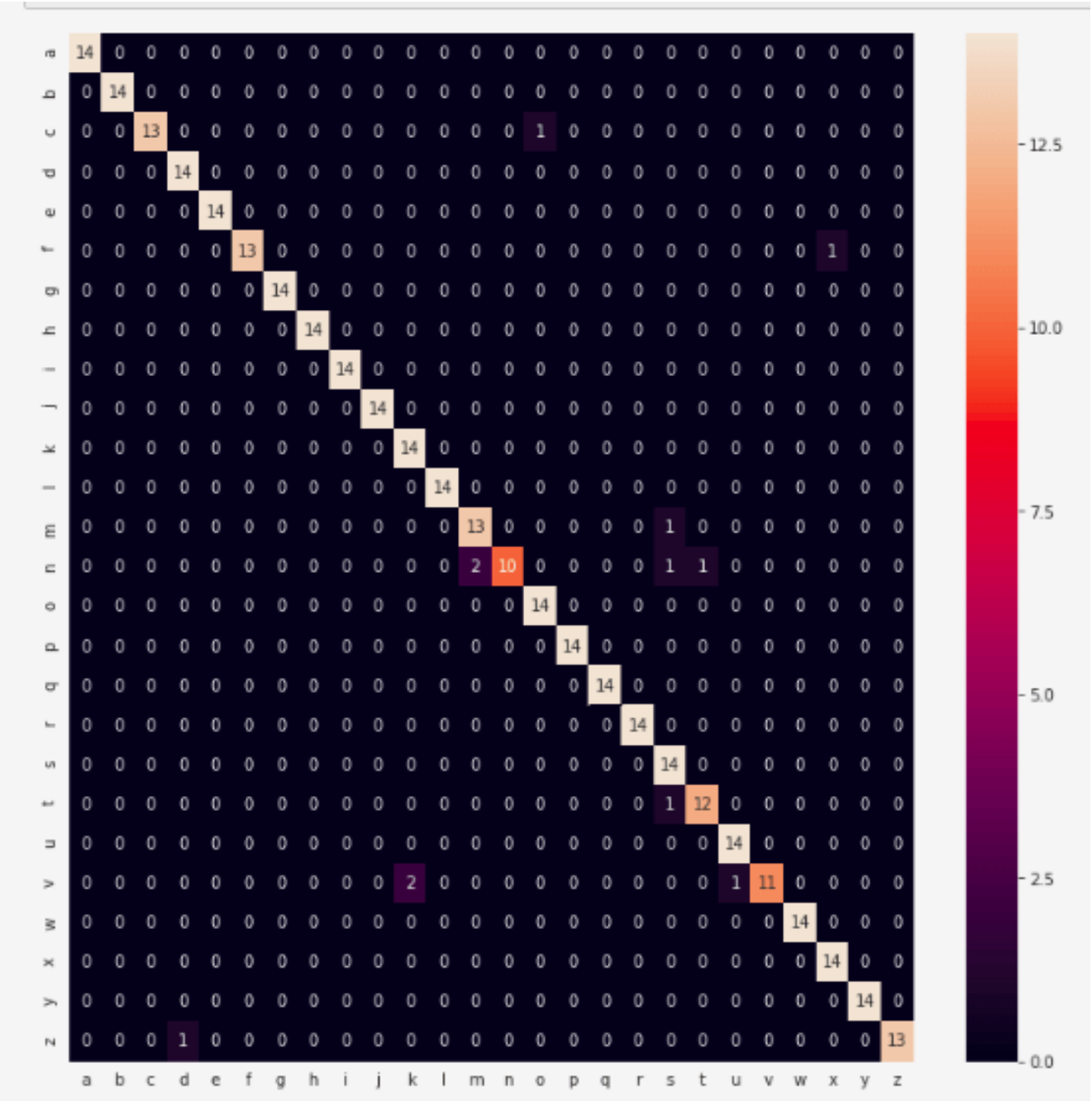
```
[[2.13186108e-10 2.13371484e-30 1.76184240e-28 ... 3.06247405e-30
  9.99998569e-01 1.13461714e-23]
 [6.93688948e-17 5.88399678e-20 1.41698770e-14 ... 3.49431645e-19
  3.18905768e-09 8.55841301e-11]
 [3.00573556e-05 8.11530481e-06 2.64593800e-05 ... 1.30931484e-02
  1.21971784e-08 1.97225305e-07]
 ...
 [3.45958458e-16 2.33072676e-13 2.38783636e-14 ... 1.96580084e-13
  2.88979073e-13 2.38219881e-03]
 [7.88769394e-05 4.25587384e-15 1.22019487e-13 ... 7.31027283e-09
  1.14031636e-05 1.52337449e-07]
 [3.02949820e-06 1.34609020e-16 8.40362563e-16 ... 9.69122699e-11
  2.47575917e-06 1.99953849e-08]]
--------------------------------
363
--------------------------------
26
--------------------------------
                Predicted classes
[24 15 13 15  6 12 23  5 18 20 24  5 18  4 25 22 15  7 13 19  2  9  3  0
 20 13 11 25  9  3  4 21 22  1 10 13  1  1  9  4 12 18 13 25  4 24 18 24
 18 25  1 23  4  1 16 18  6  2  1 18  3  1 19  7 15 23 11  8  3 20 13 17
  4 15  7 22 17  7  9  3  1  7 10  1 25  1 24  3 19 16 24 23 14 25 21  2
 11 10 10 18 13  9 14  1 18 23 10  0 11 18 15 24 17  4  2 24  0 11 21 17
  2  6  8  6 12 25 18  3 12 16  7 15 16 19 23  1 24 14  7 20 22 16 20 19
  0  6  4  3  1 18 22 21 22 25  1 17  3  2  0  5  0  7 10 22 16  3 18 21
  9  7 19 11 10 22 16 18 19 12  9  7 18  0  2  6  7 22 23 20 22 13 21 12
 21 20  6 10 10  5  6  9  8 16  8  0 20  7  5  6  2 22  7 18 12 17 14  3
  9 14 24 23 23  2 11 12 22 24 15 13  7 23 16  9 16  3 15 24 10 15  4 12
 17  0  0  8 17 15  5 25 11 20 17 13 12 19 10  0 19  4 11 16 18 25 14 21
 14 14 21 22 20 19 25 23 25  4 23  6  9 19 12 24 23  2 15 23  8 15 22 11
  0 14 11 14 25  8  0  4 20  4 23 12  4  3 17 19 17 10 24  5 14  5 14  9
  9 11  6 12 11 17 20 20  2  8 21 11  9  6 15  3  3 21 14  5 19 10 10  5
  2 10 14 16 16  2  5 14 20 16  0 20  8  5 12  8 10 12  6  8  5 17  6  8
 17  8  8]
--------------------------------
363
```

Prediction accuracy on test images:

With early stopping it achieved 96.69% accuracy on test set.

```
In [25]: acc = accuracy_score(test_labels, y_classes) * 100
         print("Accuracy on test set : {} %".format(acc))
         print('-------------------------------------------------')

         Accuracy on test set : 96.69421487603306 %
         -------------------------------------------------
```

Confusion matrix on prediction result (each class except ‘t’ in test set have 14 samples, ‘t’ has 13 samples):



Classwise prediction accuracy:

```
In [27]: classwise_prediction_accuracy = cm.diagonal()/cm.sum(axis=1)
         print("___________Class wise prediction accuracies_____________")
         dict(zip(list(string.ascii_lowercase), classwise_prediction_accuracy))

         ___________Class wise prediction accuracies_____________

Out[27]: {'a': 1.0,
          'b': 1.0,
          'c': 0.9285714285714286,
          'd': 1.0,
          'e': 1.0,
          'f': 0.9285714285714286,
          'g': 1.0,
          'h': 1.0,
          'i': 1.0,
          'j': 1.0,
          'k': 1.0,
          'l': 1.0,
          'm': 0.9285714285714286,
          'n': 0.7142857142857143,
          'o': 1.0,
          'p': 1.0,
          'q': 1.0,
          'r': 1.0,
          's': 1.0,
          't': 0.9230769230769231,
          'u': 1.0,
          'v': 0.7857142857142857,
          'w': 1.0,
          'x': 1.0,
          'y': 1.0,
          'z': 0.9285714285714286}
```

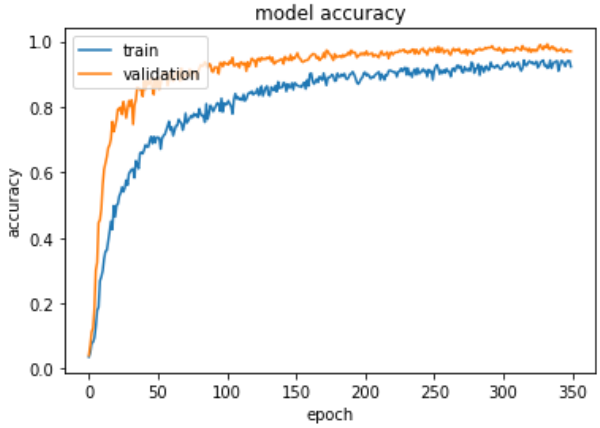
5.2 : Training without early stopping

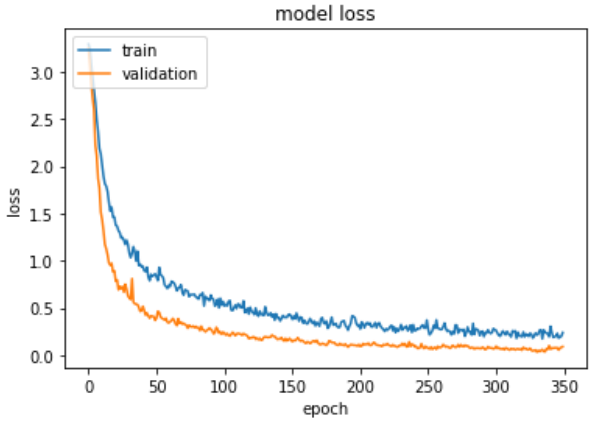
Next the classification model was trained for 350 epochs without early stopping. The model acquired 97.79% accuracy on test set.

The last 5 epochs of training log:

```
Epoch 346/350
68/68 [==============================] - 63s 930ms/step - loss: 0.2316 - acc: 0.9237 - val_loss: 0.0796 - val_ac
c: 0.9725
Epoch 347/350
68/68 [==============================] - 62s 916ms/step - loss: 0.1773 - acc: 0.9384 - val_loss: 0.0557 - val_ac
c: 0.9752
Epoch 348/350
68/68 [==============================] - 63s 922ms/step - loss: 0.1917 - acc: 0.9412 - val_loss: 0.0774 - val_ac
c: 0.9697
Epoch 349/350
68/68 [==============================] - 63s 926ms/step - loss: 0.1985 - acc: 0.9412 - val_loss: 0.0841 - val_ac
c: 0.9697
Epoch 350/350
68/68 [==============================] - 63s 932ms/step - loss: 0.2366 - acc: 0.9237 - val_loss: 0.0911 - val_ac
c: 0.9697
dict_keys(['val_acc', 'loss', 'val_loss', 'acc'])
```

Relevant curves:





Prediction accuracy on test images:

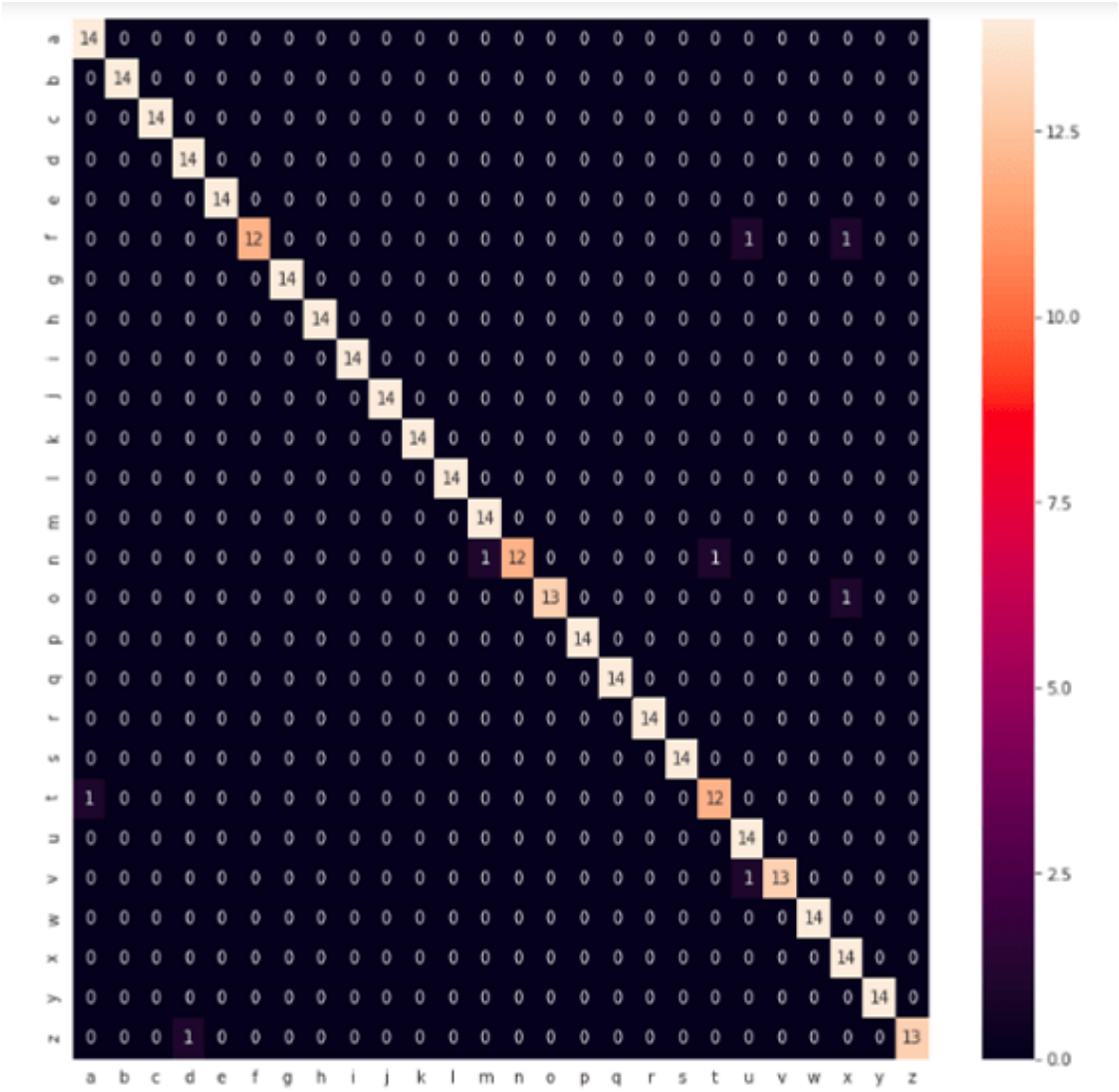
Without early stopping and a prolonged training of 350epochs, it achieved 97.79% accuracy on test set.

```
In [19]: acc = accuracy_score(test_labels, y_classes) * 100
         print("Accuracy on test set : {} %".format(acc))
         print('-------------------------------------------------')

         Accuracy on test set : 97.7961432506887 %
         -------------------------------------------------
```

Classwise prediction accuracy:

```
                  ____________Class wise prediction accuracies_____________

Out[21]: {'a': 1.0,
          'b': 1.0,
          'c': 1.0,
          'd': 1.0,
          'e': 1.0,
          'f': 0.8571428571428571,
          'g': 1.0,
          'h': 1.0,
          'i': 1.0,
          'j': 1.0,
          'k': 1.0,
          'l': 1.0,
          'm': 1.0,
          'n': 0.8571428571428571,
          'o': 0.9285714285714286,
          'p': 1.0,
          'q': 1.0,
          'r': 1.0,
          's': 1.0,
          't': 0.9230769230769231,
          'u': 1.0,
          'v': 0.9285714285714286,
          'w': 1.0,
          'x': 1.0,
          'y': 1.0,
          'z': 0.9285714285714286}
```

Confusion matrix on prediction result (each class except 't' in test set have 14 samples, 't' has 13 samples):



Conclusion:

With early stopping the CNN model gave reasonably good **96.69% accuracy** on test set. After training the model for 350 epochs, **without early stopping and after a prolonged training of 350 epochs,** the model gave slightly better accuracy, **97.79%**.