Here is my interpretation of my model.
The jupiter notebook is already in my github
https://github.com/A01706648/tc3002b_ml

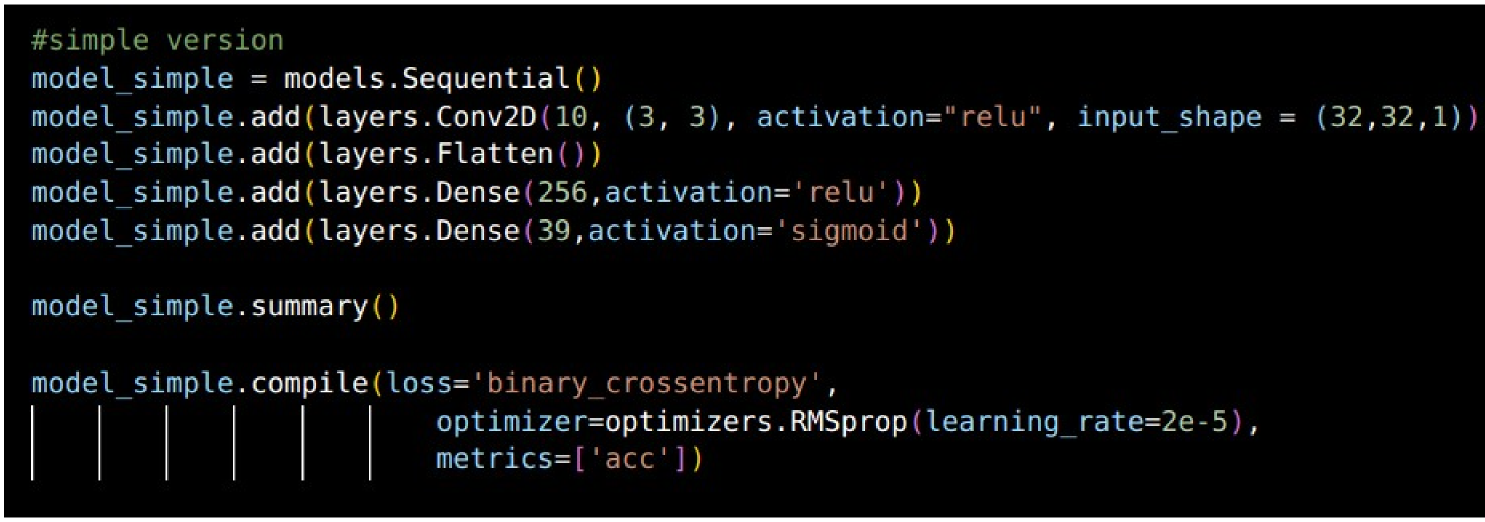
1, Introduction
The project is training a module to recognize the hand writing character.
It is with 39 symbols.

2, Dataset Split
training data set is with 800K+ images,(32x32 pixels)
testing data set is with 22K+ image, (32x32 pixels)

Due to the huge training data set, and limited time,
I took 5% of the original training data set as real training data set,
and took 10% of the real training data set as validation data set.
Then took the complete testing data set for final testing.

Meanwhile,
I use the ImageDataGenerator to create argumentation dataset,
to enrich the training dataset.

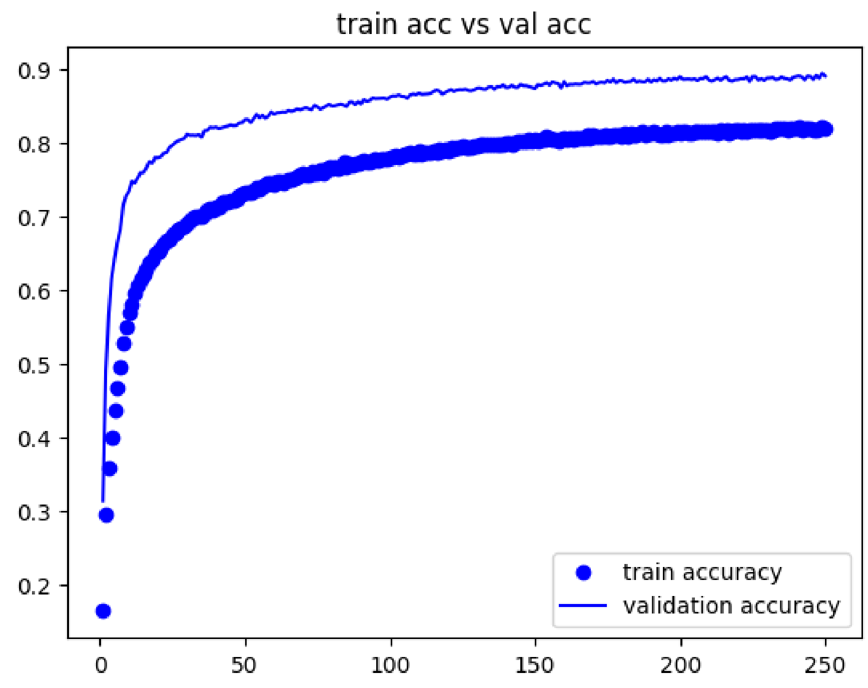
3, Simple Model
I create a simple model initially,
with one Conv2D and a 256 neural network

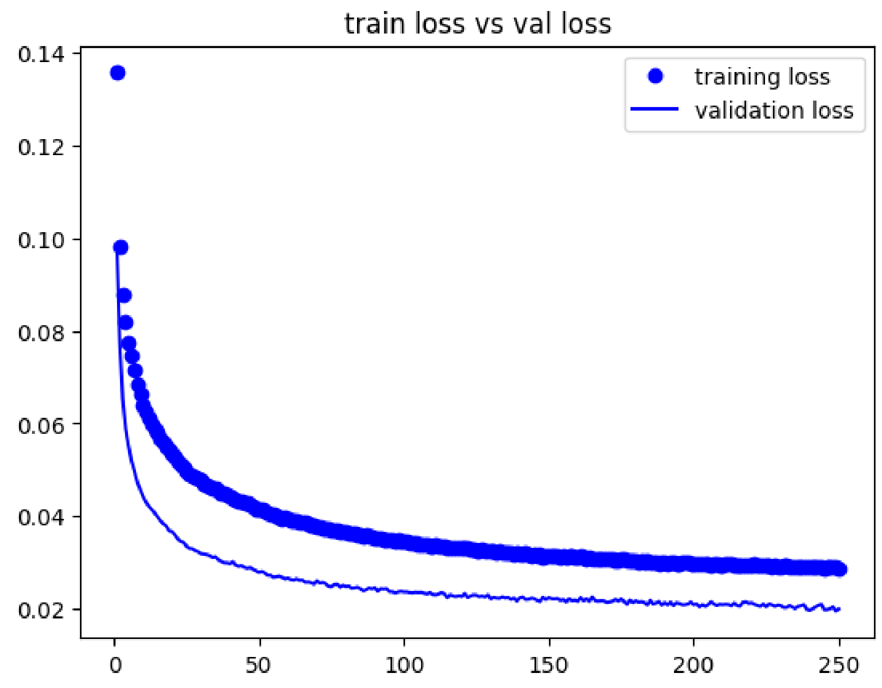


```
#simple version
model_simple = models.Sequential()
model_simple.add(layers.Conv2D(10, (3, 3), activation="relu", input_shape = (32,32,1))
model_simple.add(layers.Flatten())
model_simple.add(layers.Dense(256,activation='relu'))
model_simple.add(layers.Dense(39,activation='sigmoid'))

model_simple.summary()

model_simple.compile(loss='binary_crossentropy',
                     optimizer=optimizers.RMSprop(learning_rate=2e-5),
                     metrics=['acc'])
```

I did a 250 epoches training with training data and validation data
The training accuracy is increasing steadily, meanwhile validation accuracy is increasing steadily too.
in the same time, training loss and validation loss is decreasing.

So we can conclude the model is traing properly, and WITHOUT over fit.



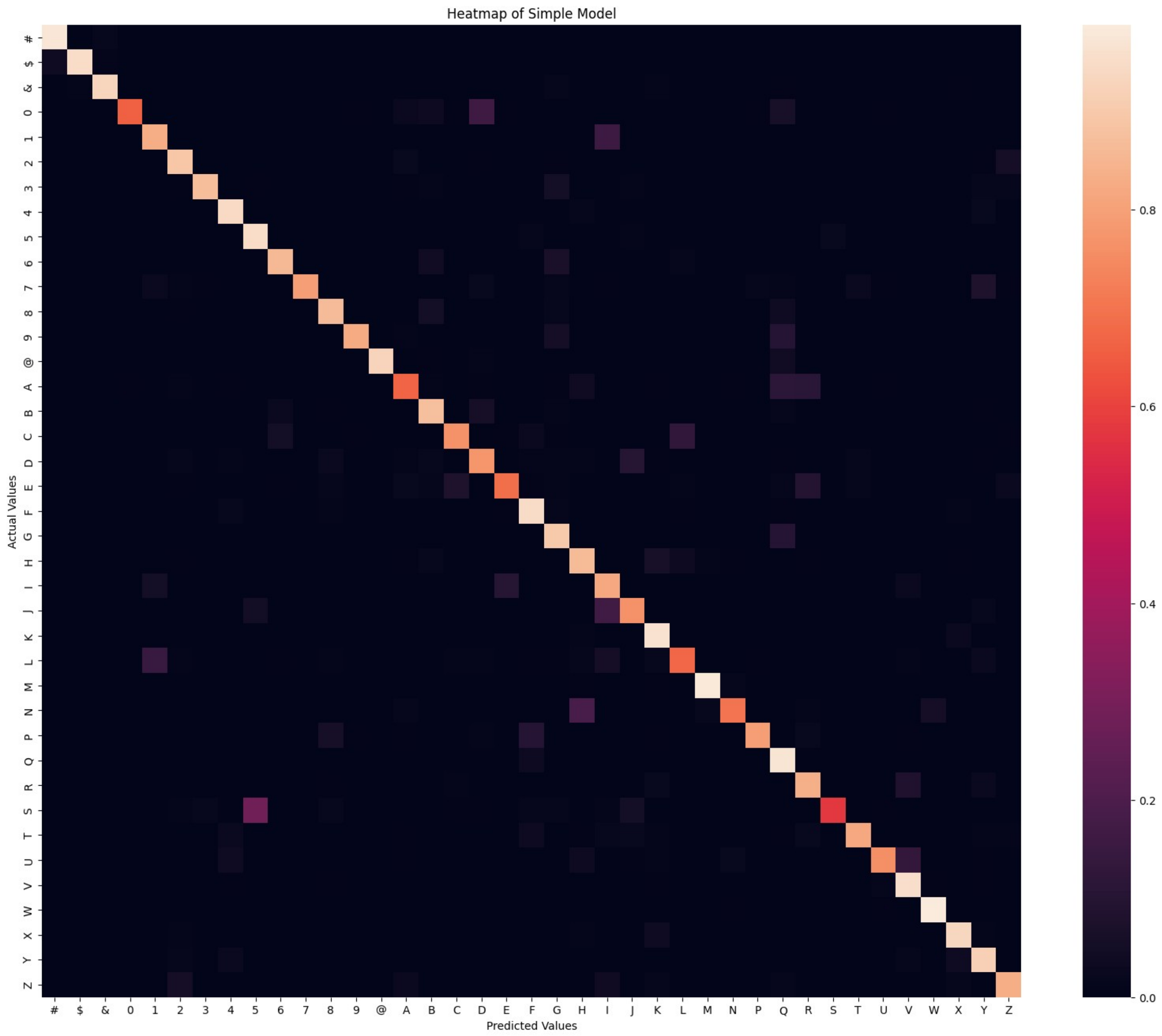


After test with the testing data,
we get 86.75% accuracy

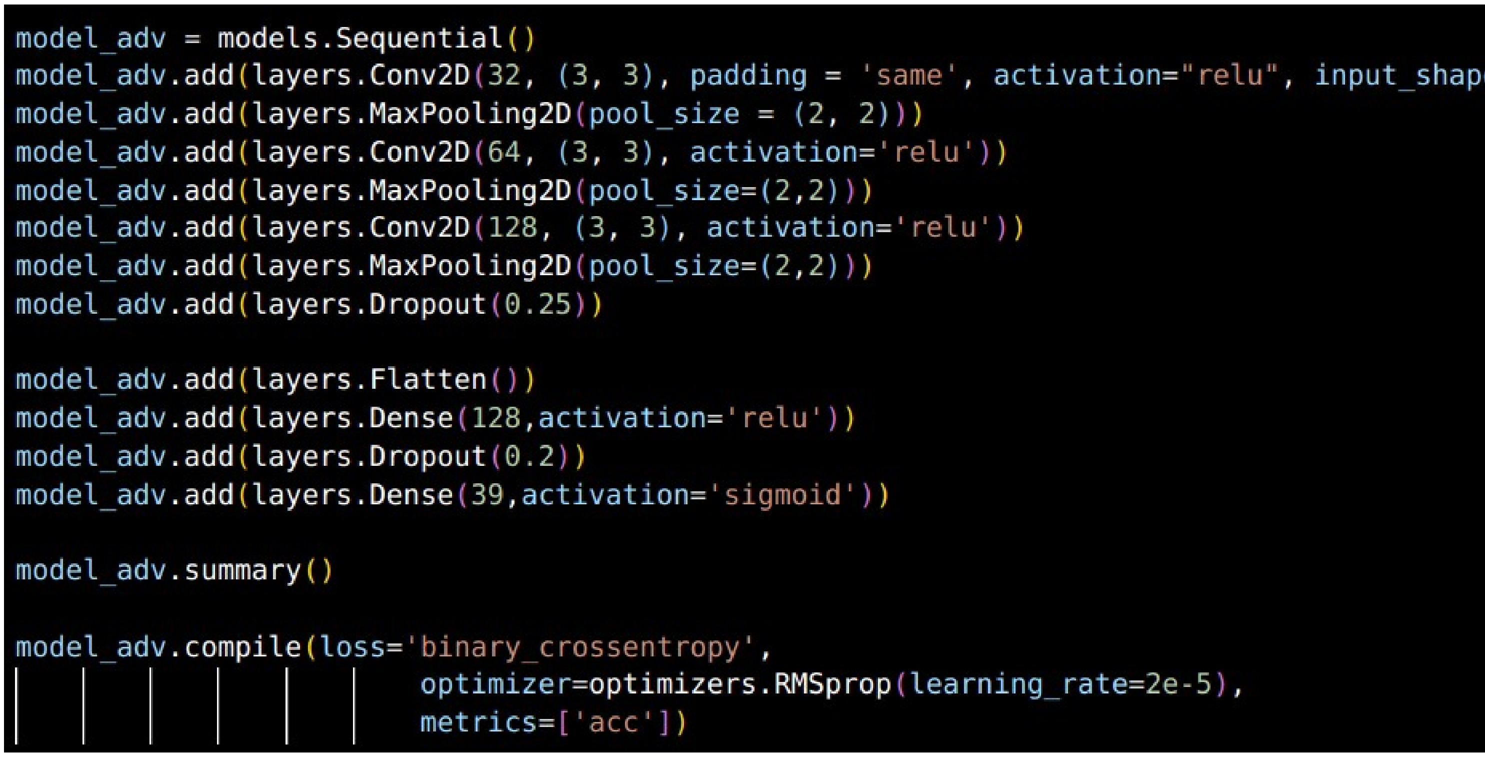
With the confussion map
We can notice
0 is confused with D
1 is confused with L
A is confused with Q and R
C is confused with L
E is confused with R

L is confused with 1
S is confused with 5 a lot
U is confused with V

In the context of writing,
It is very command to mixed 0 with D, 1 with L, S with 5, and U with V
But it is not common to mix A with R and Q, C mix with L, and E mixed with R.
So we need to improve this module



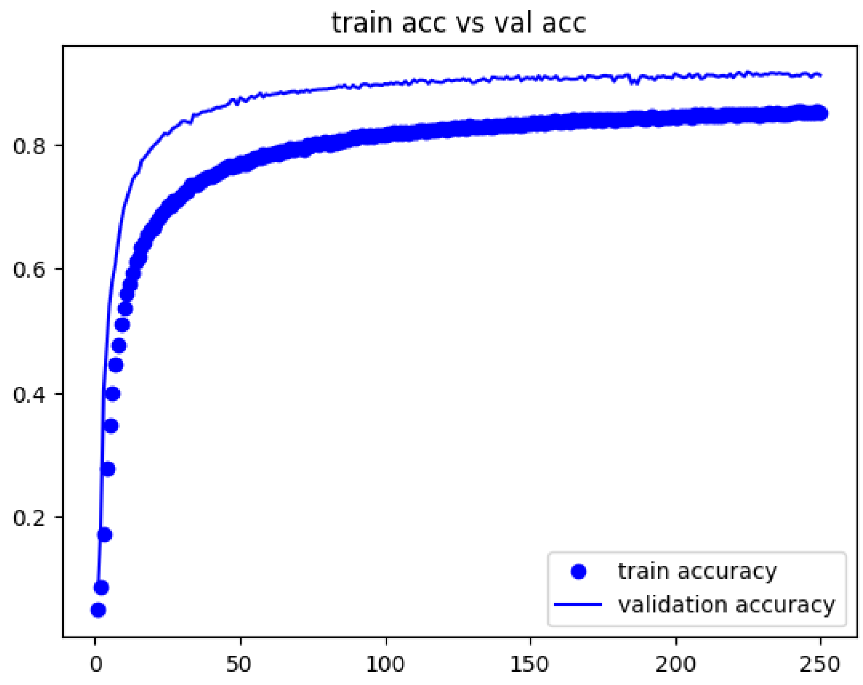
5, Advanced Model,
In order to get better accuracy,
I used 3 layer of Conv2D and one 128 neutral network

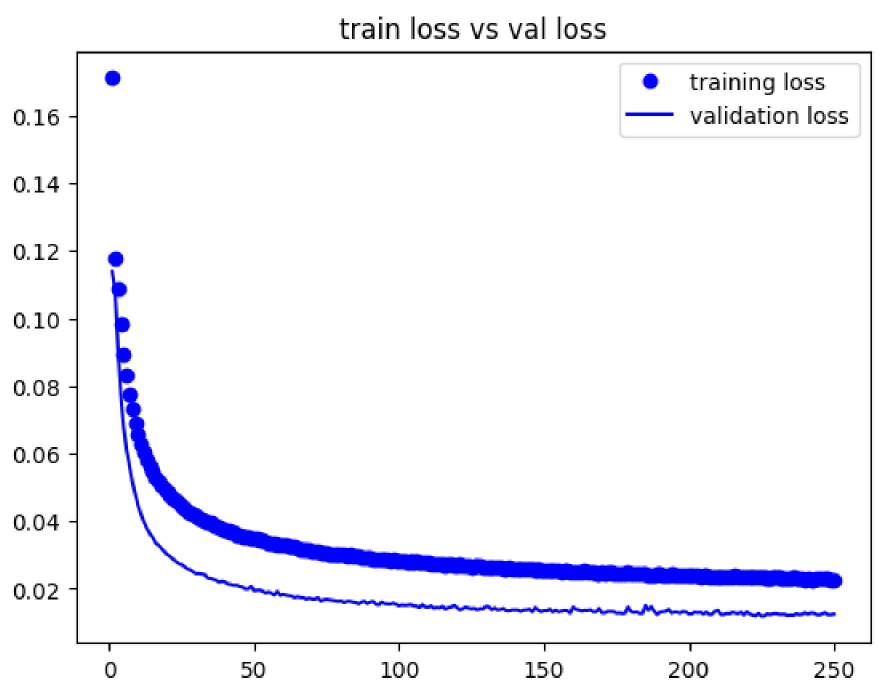


```
model_adv = models.Sequential()
model_adv.add(layers.Conv2D(32, (3, 3), padding = 'same', activation="relu", input_shap
model_adv.add(layers.MaxPooling2D(pool_size = (2, 2)))
model_adv.add(layers.Conv2D(64, (3, 3), activation='relu'))
model_adv.add(layers.MaxPooling2D(pool_size=(2,2)))
model_adv.add(layers.Conv2D(128, (3, 3), activation='relu'))
model_adv.add(layers.MaxPooling2D(pool_size=(2,2)))
model_adv.add(layers.Dropout(0.25))

model_adv.add(layers.Flatten())
model_adv.add(layers.Dense(128,activation='relu'))
model_adv.add(layers.Dropout(0.2))
model_adv.add(layers.Dense(39,activation='sigmoid'))

model_adv.summary()

model_adv.compile(loss='binary_crossentropy',
                        optimizer=optimizers.RMSprop(learning_rate=2e-5),
                        metrics=['acc'])
```

I did a 250 epoches training with training data and validation data
The training accuracy is increasing steadily, meanwhile validation accuracy is increasing steadily too.
in the same time, training loss and validation loss is decreasing.

So we can conclude the model is traing properly, and WITHOUT over fit.



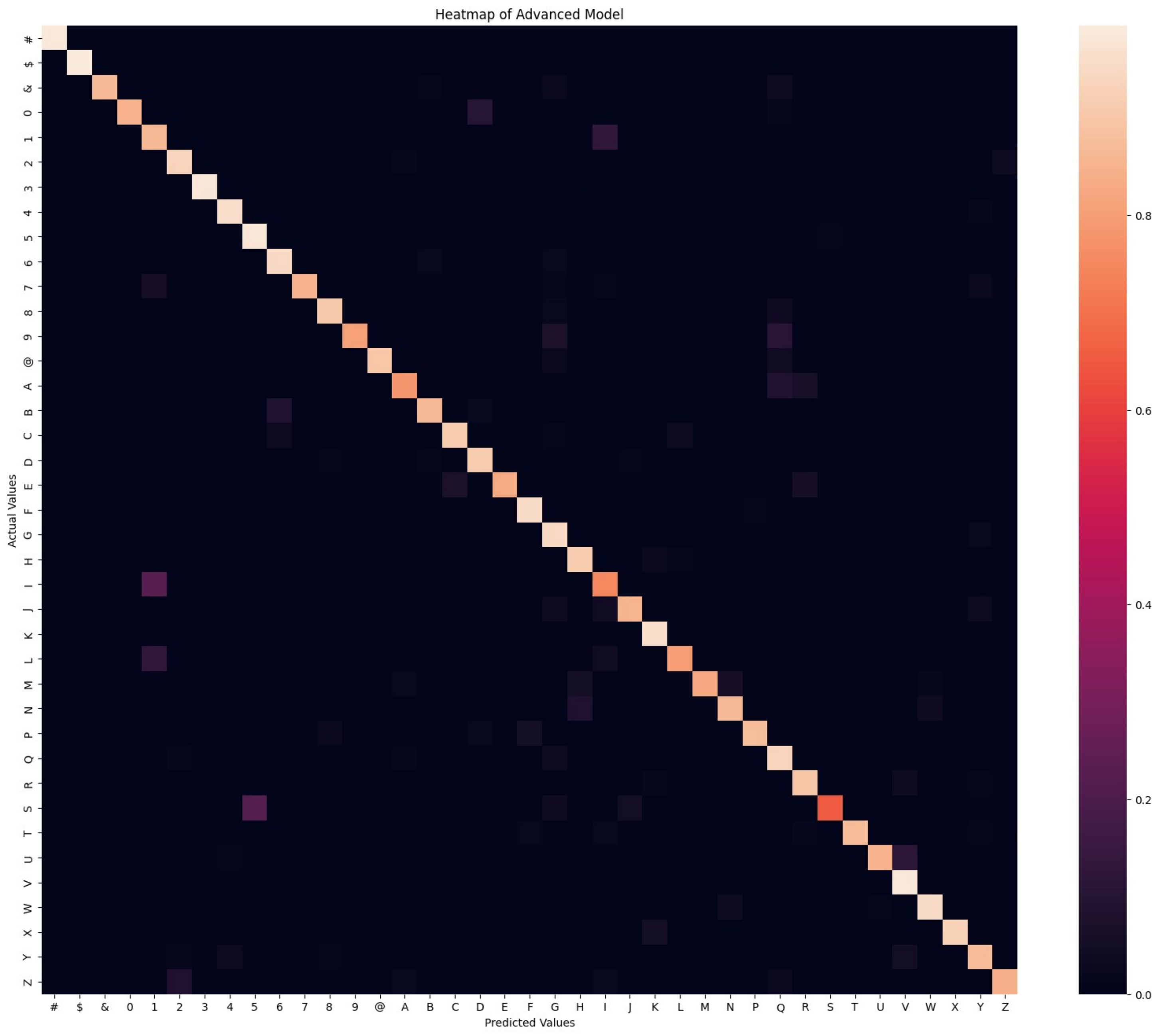


After test with the testing data,
we get 90.99% accuracy, which is 4.24% higher than the simple module.

With the confusion table.
We can notice the accuracy has improved a lot.
It improved 0, A, E

But still remain some problem
I confused with 1
S confused with 5

This module have improved a lot,
in the context of write, even human can not different the I with 1 and S with 5,
so it is fine for this module


Heatmap of Advanced Model
Actual Values
Predicted Values
# $ & 0 1 2 3 4 5 6 7 8 9 @ A B C D E F G H I J K L M N P Q R S T U V W X Y Z
0.0
0.2
0.4
0.6
0.8

6, Conclusion

With our improved the module, we still keep the right fit of the module,
and able to reach around 91% accuracy,
And reduced several character's error rate.