Wenguang Hu A01706648

End of Sem 2: Generation or selection of data set 1%

Get, generate or augment a data set.

I created a augment generate of the training data.
Due to the training data is the handwriting data, So I will only slight rotate it, and do not shift or zoom too much, because I want to keep the writing in the picture completely.
In addition, the writing is NOT horizontal symmetrical, so I will NOT do horizontal flip

```
train_datagen = ImageDataGenerator(
                    rescale = 1./255,
                    rotation_range = 15,
                    width_shift_range = 0.1,
                    height_shift_range = 0.1,
                    shear_range = 0.3,
                    zoom_range = 0.3,
                    horizontal_flip = False,)
```

Make the separation of the test and training sets.

The training and test set is already separated in the data set.
The are stayed in two different folder “Train” and “Test”.

But due to the huge number of the Training Data Set, I decide to take 5% of the Data for training.
So I create a folder called “TrainSmall”, and move 5% of the data set from “Train” folder to “TrainSmall” folder

```
#create smaller train folder
os.mkdir(train_small_dir)

dir_list = os.listdir(train_orig_dir)
for symb_dir in dir_list:
    train_orig_symb_dir = os.path.join(train_orig_dir, symb_dir)
    train_small_symb_dir = os.path.join(train_small_dir, symb_dir)

    #select the val file
    train_orig_file_list = os.listdir(train_orig_symb_dir)
    file_num = len(train_orig_file_list)
    train_small_file_num = math.floor(file_num * train_small_radio)
    train_small_file_list = random.sample(train_orig_file_list, train_small_file_num)
    #print(symb_dir)
    #print(val_file_list)

    #create symbol dir in validation folder
    os.mkdir(train_small_symb_dir)
    #cp selected file to val symbol dir
    for train_small_file in train_small_file_list:
        train_file_orig_path = os.path.join(train_orig_symb_dir, train_small_file)
        shutil.copy(train_file_orig_path, train_small_symb_dir)
```

(variable) train_orig_symb_dir: str

In addition, I did do my separation of Training Data Set with Validation Data Set.
I create a ‘Validation’ Folder, and move 10% of the random Data Set from “TrainSmall” to “Validation Folder”

```
#create validation folder
os.mkdir(validation_dir)

dir_list = os.listdir(train_dir)
for symb_dir in dir_list:
    train_symb_dir = os.path.join(train_dir, symb_dir)
    val_symb_dir = os.path.join(validation_dir, symb_dir)

    #select the val file
    train_file_list = os.listdir(train_symb_dir)
    file_num = len(train_file_list)
    val_file_num = math.floor(file_num * validation_radio)
    val_file_list = random.sample(train_file_list, val_file_num)
    #print(symb_dir)
    #print(val_file_list)

    #create symbol dir in validation folder
    os.mkdir(val_symb_dir)
    #move selected file to val symbol dir
    for val_file in val_file_list:
        val_file_orig_path = os.path.join(train_symb_dir, val_file)
        shutil.move(val_file_orig_path, val_symb_dir)
```

End of Sem 2: Pre-processing of data 1%

Apply scaling techniques.

Do the relevant preprocessing of the data

I did scale down all the picture to 0~1 scale, by divided the pixel value by 255.
And the picture is already a clean gray value picture, no noise on it, so I do not need to do preprocess on it. And all the picture is already in 32x32 size, so I do not need to change the size of them

```
train_datagen = ImageDataGenerator(
                    rescale = 1./255,
                    rotation_range = 15,
                    width_shift_range = 0.1,
                    height_shift_range = 0.1,
                    shear_range = 0.3,
                    zoom_range = 0.3,
                    horizontal_flip = False,)
```

```
validation_datagen = ImageDataGenerator(rescale = 1./255)
```

```
train_generator = train_datagen.flow_from_directory(
                    train_dir,
                    target_size = (32, 32),
                    batch_size = batch_size,
                    class_mode ='categorical',
                    color_mode="grayscale",
```

End of Sem 3: Model Implementation 1%

Select a model supported by a state of the art paper.

I use one CNN layer, then followed with one MLP layer, to do the classification.

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_12 (Conv2D)          (None, 30, 30, 10)        100

 flatten_6 (Flatten)         (None, 9000)              0

 dense_12 (Dense)            (None, 256)               2304256

 dense_13 (Dense)            (None, 39)                10023

=================================================================
Total params: 2,314,379
Trainable params: 2,314,379
Non-trainable params: 0
```

Implement the model using a selected framework.

I am using tensorflow and kera to do the module implementation

```
from tensorflow.keras import optimizers
from tensorflow.keras import models
from tensorflow.keras import layers
```

```
#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'])
```

End of Sem 3: Initial evaluation of the model 1%

Select suitable metrics backed by a state of the art paper.

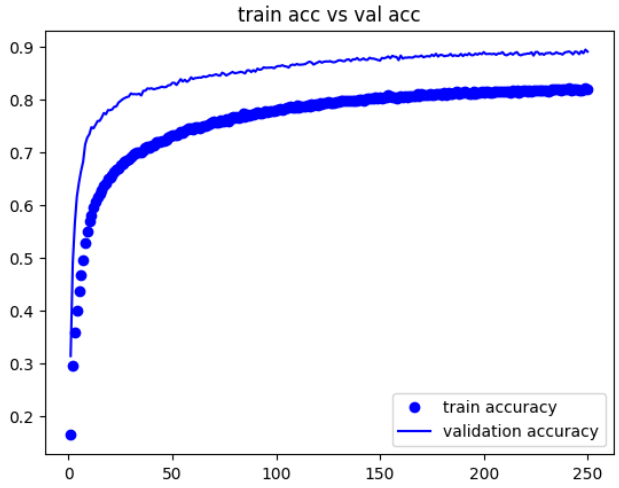
I use the test accuracy as the metrics for the training.
Since it is a classifier, so accuracy shall be a suitable metrics for the model.

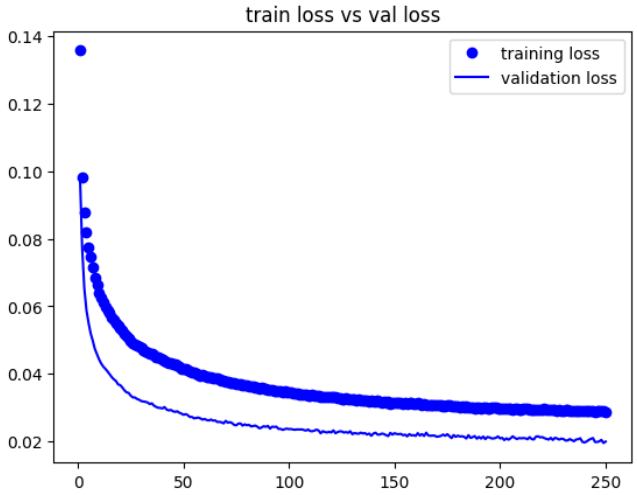
```
model_simple.summary()

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

Report results obtained and interpret them.

I did a 250 epochs 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 trained properly, and WITHOUT over fitting





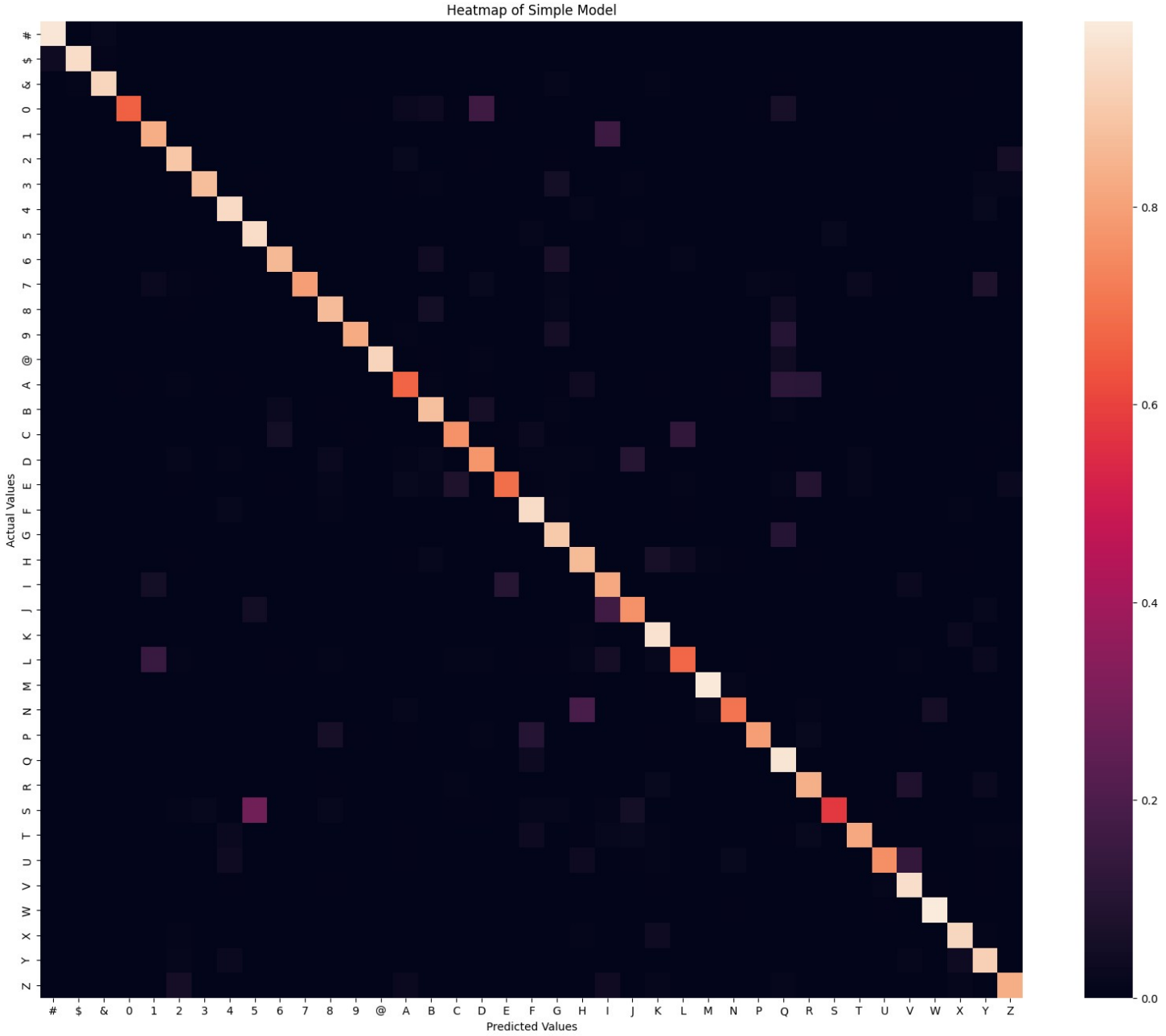
.

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

End of Sem 4: Model Refinement 1%

Measure the performance of the current model

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

Adjust model or architecture hyper parameters

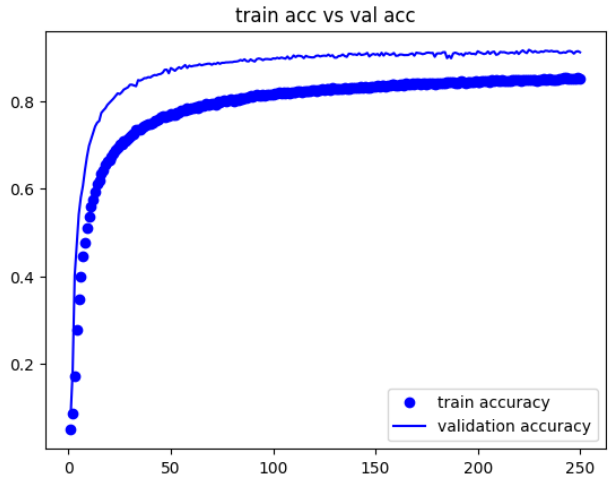
In order to get better accuracy,
I used 3 layer of Conv2D and one 128 neutral network

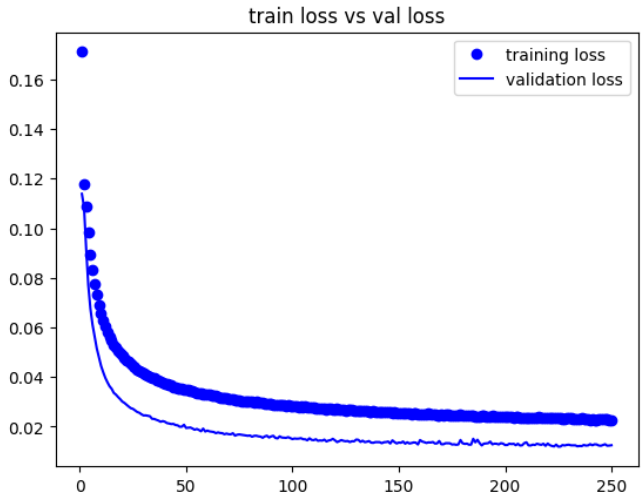
```
model_adv = models.Sequential()
model_adv.add(layers.Conv2D(32, (3, 3), padding = 'same', activation="relu", input_shap
model_adv.add(layers.MaxPooling2D(pool_size = (2, 2)))
model_adv.add(layers.Conv2D(64, (3, 3), activation='relu'))
model_adv.add(layers.MaxPooling2D(pool_size=(2,2)))
model_adv.add(layers.Conv2D(128,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

Compare to previous performance and report improvements

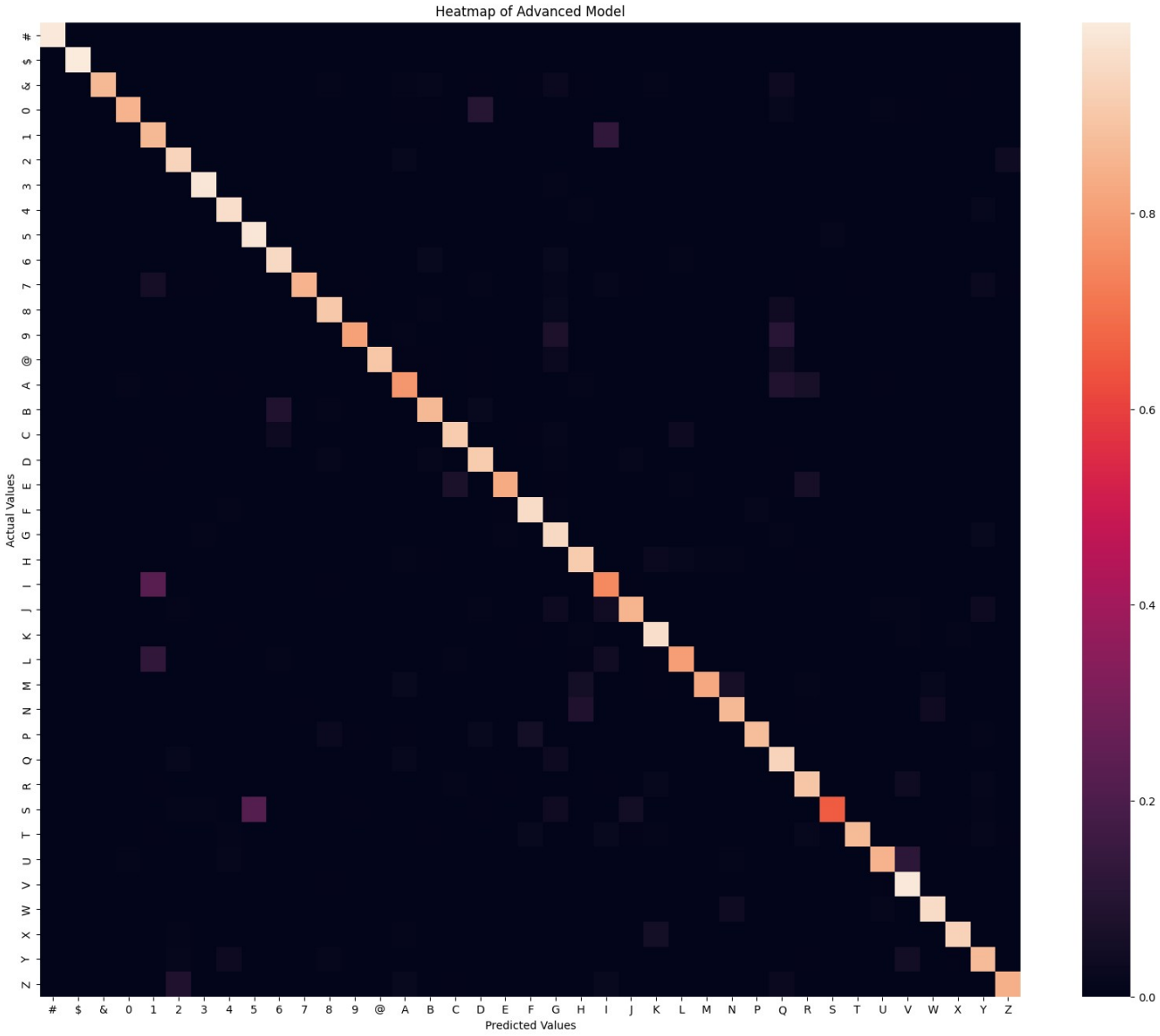
I did a 250 epoches training with training data and validation data
The training accuracy is increasing steadily, meanwhile validation accuracy is increasing steadily too.
in the same time, training loss and validation loss is decreasing.

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