Project 2 Deep Digit Image Recognition
CMSC 691 Computer Vision
Spring 2022
Due 03 / 29 / 2022

You must create a convolutional neural network (CNN) to classify MNIST images into digits from 0 to 9. Your neural network architecture must be a ResNet5 like architecture as specified in the description. You must use a custom training loop with minibatch size 100.

Label 7

Label 2

Label 1

A ResNet architecture has skip connections between each pair of layers.   We must implement a specific resnet-5 like architecture. You must create this architecture from scratch using the Keras functional model.   The details of this architecture are shown in the below diagram.

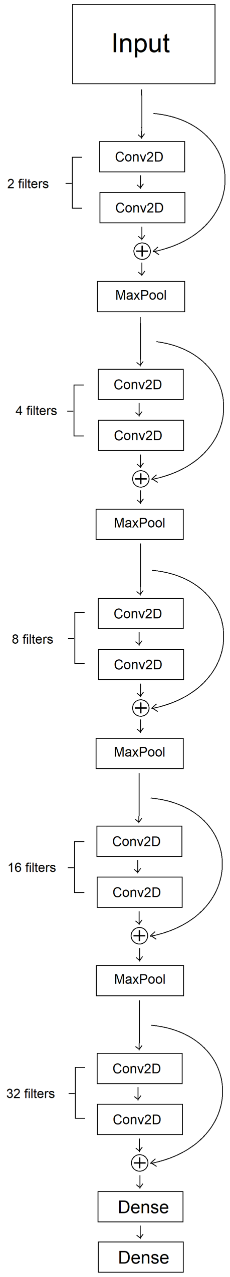
NOTES

Model architecture must be the ResNet-5 as seen to the right

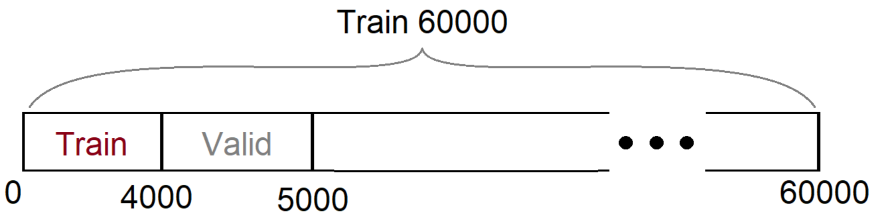
⊕ layers are the concatenation of two vectors

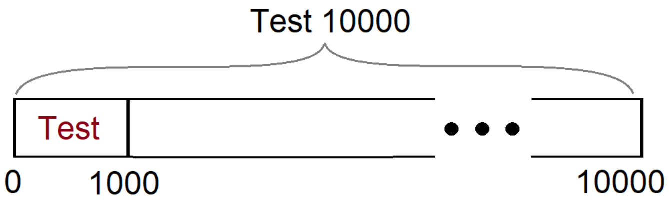
Conv2D must have kernel 3

MaxPool must be (2,2) with stride 2

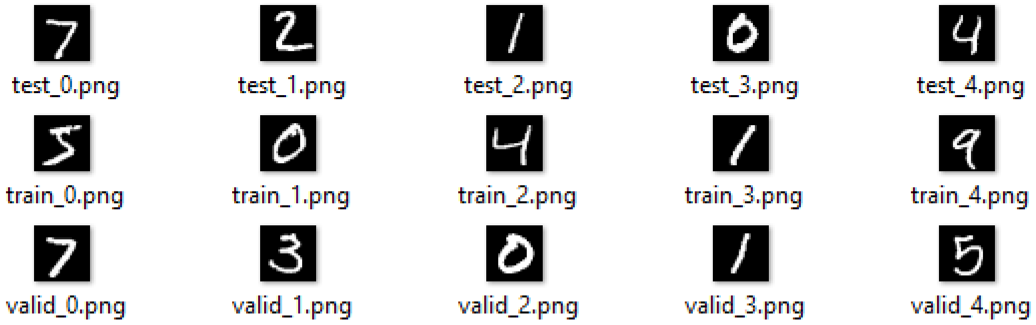


You must separate the training data into train, validation and test splits, the MNIST data has 60000 training images, and 10000 test images. You must extract 4000 training, 1000 validation, and 1000 test images.





Your program must create a directory called "output" and you must save the first 5 characters of each test, test, and validation split as follows. Your output must identically match the following,



<u>You must create a custom training loop</u>, and train the model using the keras gradientTape.

You must use GradientTape

You cannot use *model.fit*

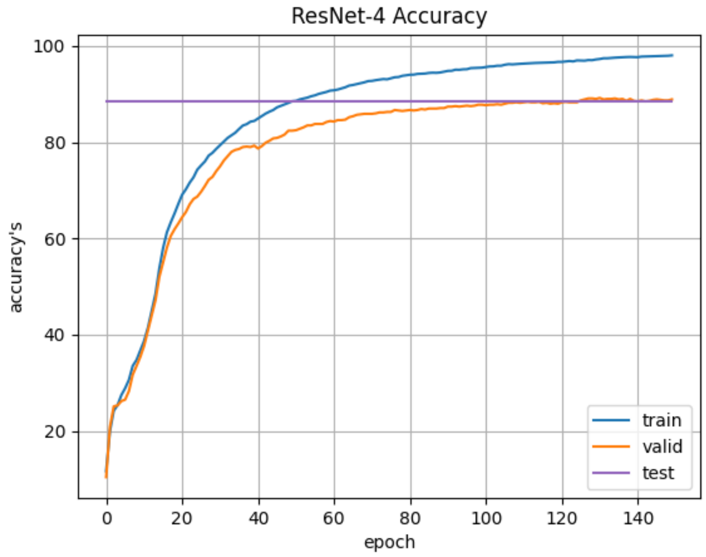
Your loop must be of the following form

```
for epoch in range(num_epochs):
    for batch in range(num_batch):
        # perform gradient update
```

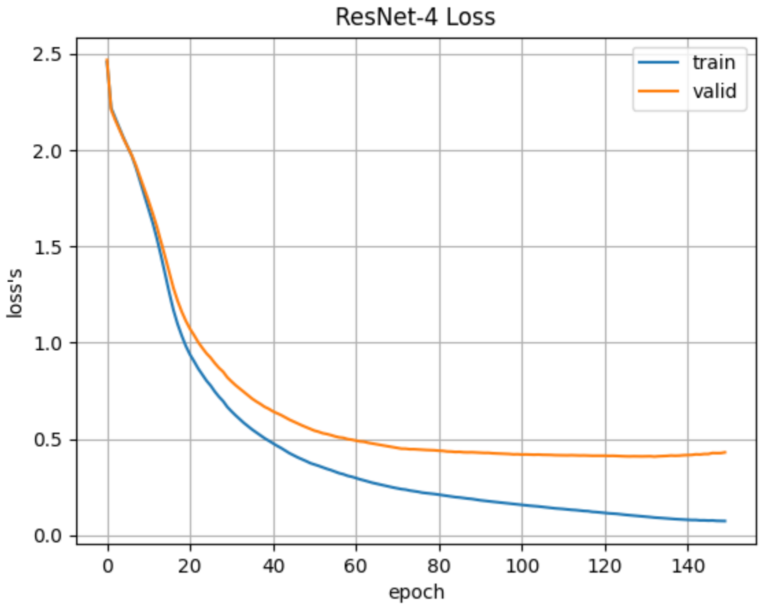
You must implement the log-loss functional

$$LogLoss = -1/N \sum_{i=1}^{N} \sum_{j=1}^{M} Y_{ij} \log(F(X_{ij}))$$

You must plot your your training, validation, and test accuracy and save them to the folder “output”

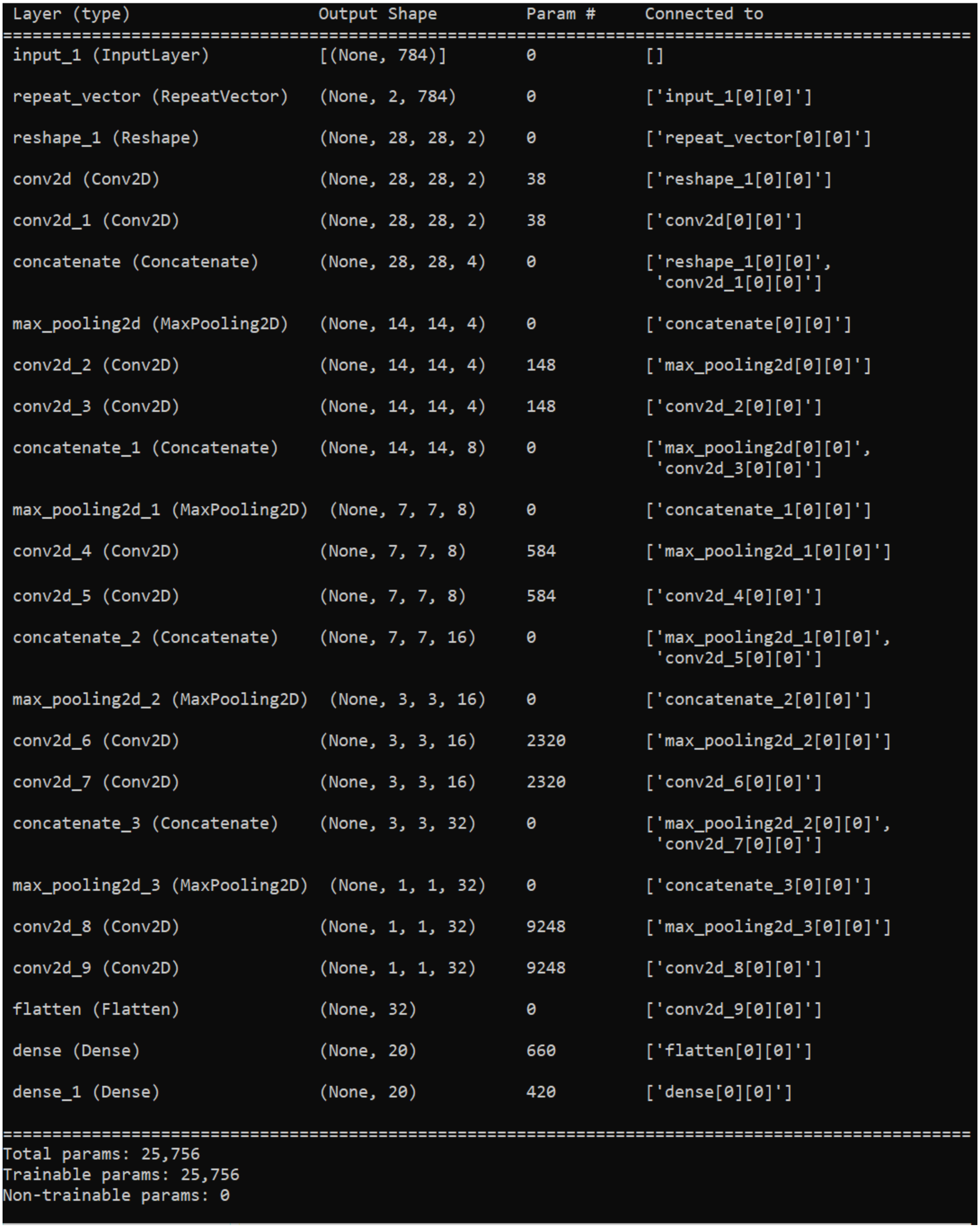


output/accuracy.png



output/loss.png

You must print the model summary to the console() prior to training the model



```
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 784)]        0           []

 repeat_vector (RepeatVector)   (None, 2, 784)       0           ['input_1[0][0]']

 reshape_1 (Reshape)            (None, 28, 28, 2)    0           ['repeat_vector[0][0]']

 conv2d (Conv2D)                (None, 28, 28, 2)    38          ['reshape_1[0][0]']

 conv2d_1 (Conv2D)              (None, 28, 28, 2)    38          ['conv2d[0][0]']

 concatenate (Concatenate)      (None, 28, 28, 4)    0           ['reshape_1[0][0]',
                                                                  'conv2d_1[0][0]']

 max_pooling2d (MaxPooling2D)   (None, 14, 14, 4)    0           ['concatenate[0][0]']

 conv2d_2 (Conv2D)              (None, 14, 14, 4)    148         ['max_pooling2d[0][0]']

 conv2d_3 (Conv2D)              (None, 14, 14, 4)    148         ['conv2d_2[0][0]']

 concatenate_1 (Concatenate)    (None, 14, 14, 8)    0           ['max_pooling2d[0][0]',
                                                                  'conv2d_3[0][0]']

 max_pooling2d_1 (MaxPooling2D)  (None, 7, 7, 8)     0           ['concatenate_1[0][0]']

 conv2d_4 (Conv2D)              (None, 7, 7, 8)      584         ['max_pooling2d_1[0][0]']

 conv2d_5 (Conv2D)              (None, 7, 7, 8)      584         ['conv2d_4[0][0]']

 concatenate_2 (Concatenate)    (None, 7, 7, 16)     0           ['max_pooling2d_1[0][0]',
                                                                  'conv2d_5[0][0]']

 max_pooling2d_2 (MaxPooling2D)  (None, 3, 3, 16)    0           ['concatenate_2[0][0]']

 conv2d_6 (Conv2D)              (None, 3, 3, 16)     2320        ['max_pooling2d_2[0][0]']

 conv2d_7 (Conv2D)              (None, 3, 3, 16)     2320        ['conv2d_6[0][0]']

 concatenate_3 (Concatenate)    (None, 3, 3, 32)     0           ['max_pooling2d_2[0][0]',
                                                                  'conv2d_7[0][0]']

 max_pooling2d_3 (MaxPooling2D)  (None, 1, 1, 32)    0           ['concatenate_3[0][0]']

 conv2d_8 (Conv2D)              (None, 1, 1, 32)     9248        ['max_pooling2d_3[0][0]']

 conv2d_9 (Conv2D)              (None, 1, 1, 32)     9248        ['conv2d_8[0][0]']

 flatten (Flatten)              (None, 32)           0           ['conv2d_9[0][0]']

 dense (Dense)                  (None, 20)           660         ['flatten[0][0]']

 dense_1 (Dense)                (None, 20)           420         ['dense[0][0]']

==================================================================================================
Total params: 25,756
Trainable params: 25,756
Non-trainable params: 0
```

Grading Rubric

Requirements	**Deductions**
Code must be written in Python using Keras (Tensorflow 2)	-100 pts
Code must be a single python file named proj2.py	-100 pts
Code must use only the following dependencies tensorflow2 matplotlib opencv numpy	-100 pts
Python file must run straight through to completion without user input	-75 pts
Architecture must be ResNet-5 nearly identical as described in above architecture diagram	-75 pts
Training must use a custom loop with gradient tape	-75 pts
First 5 images of train / test / validation split must be written to output directory	-30 pts
Accuracy (with train validation and test) and Loss (with train and validation) must be saved to accuracy.png and loss.png	-30 pts
Architecture must implement the log-loss function	-15 pts
Script must attempt to create folder “output”, and must not crash if output folder already exists	-30 pts
Architecture must learn and achieve acceptable accuracy	-50 pts