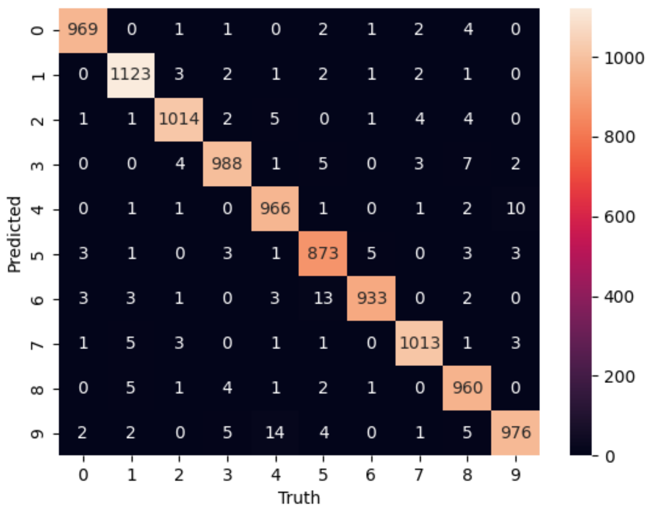
This assignment focused on low rank model compression. In order to achieve compression singular value decomposition is used to isolate a lowrank model from the resulting matrices where the origin matrix is each set of weights from a previously trained model. The derived weights are used to initialize a new compressed model with reduced parameters which preserves the accuracy of the original model in decreasing amounts as the compression is increased.

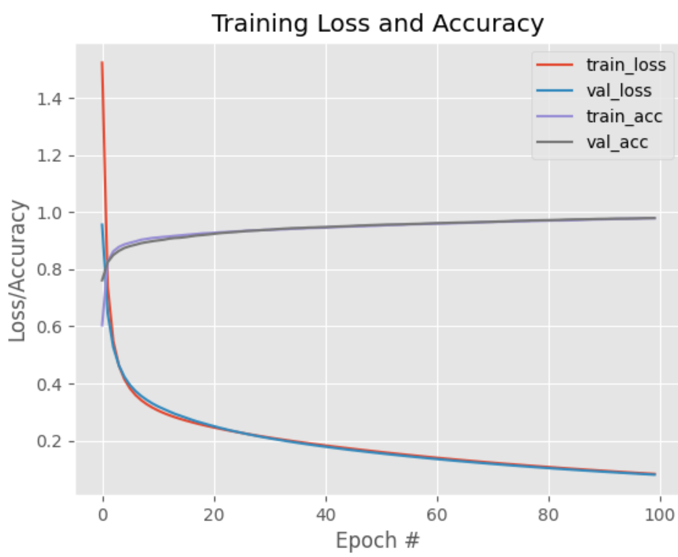
Step 1: Design and Train the light-weight model: The model created is a dense feed forward neural network with 100, 50, and 10 neuron layers respectively. Here's the model summary. Layers are fully connected and contain a bias neuron initiated to 1. Relu activations are used.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)            Output Shape          Param #
=================================================================
 flatten (Flatten)        (None, 784)          0

 dense (Dense)            (None, 100)          78500

 dense_1 (Dense)          (None, 50)           5050

 dense_2 (Dense)          (None, 10)           510

=================================================================
Total params: 84,060
Trainable params: 84,060
Non-trainable params: 0
```

To train this network the mnist dataset was used. To preprocess the images they were shuffled, converted to floats, normalized to 0-1, translated in the x and y axis between -3 and +3, as well as rotated randomly between +35 and -35 degrees to increase the robustness of the model. Below you'll find the results of this model trained on the mnist dataset. Results were achieved using a .01 learning rate over 100 epochs.



Epoch 100/100
79/79 [==============================] - 0s 4ms/step - loss: 0.0833 - accuracy: 0.9791 - val_loss: 0.0806 - val_accuracy: 0.9795



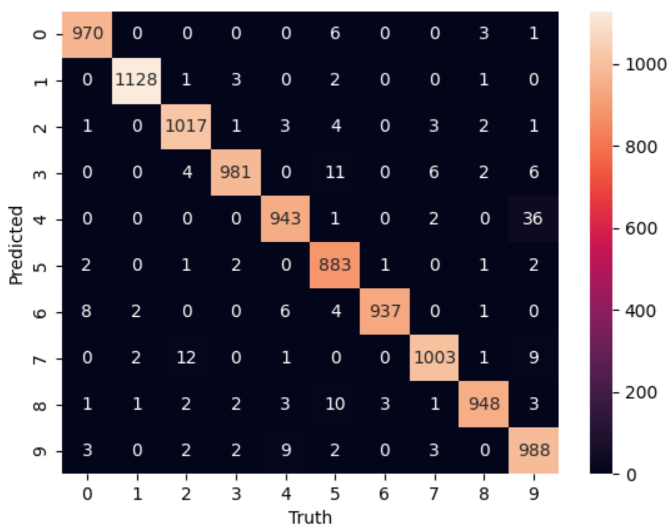
Step 2: Generate the low rank model:

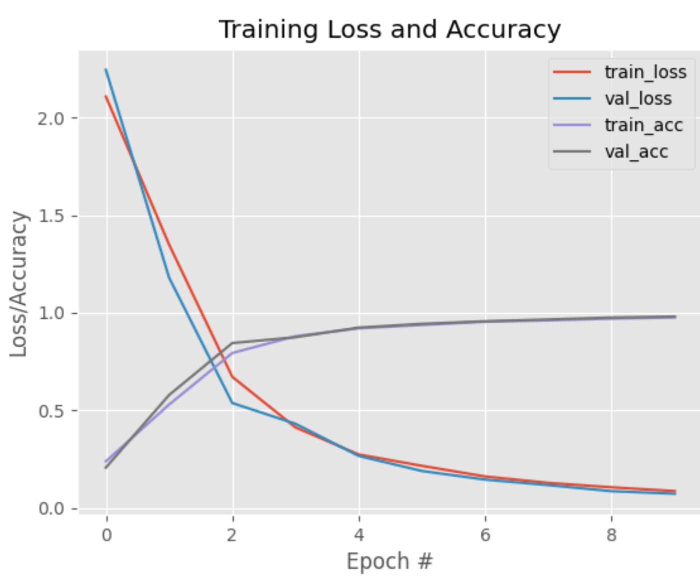
To generate the low rank model I took advantage of the tensorflow linear algebra library to compute svd on each model layer. Following this the values were converted to numPy floats and scaled by the compression value set earlier in the program (for this section it was set to a default value of 1 for no compression). This resulted in u e vT, the singular alue decomposition of the weight matrix. For simplification u and e were multiplied into a single matrix and stored to be later used as weight layers in the new model. To create the new model I set each layer to the first shape dimension given a shape (x,y). From there I used the weights and bias from the previous steps. Some bias layers needed to be initiated randomly due to the change in shape, whereas for the U' layers the arrays could be sliced along with the matrix decomposition to match. This strategy seems a little hand wavey without running more testing.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)           Output Shape          Param #
=================================================================
 flatten (Flatten)       (None, 784)          0

 dense (Dense)           (None, 100)          78500

 dense_1 (Dense)         (None, 100)          10100

 dense_2 (Dense)         (None, 50)           5050

 dense_3 (Dense)         (None, 50)           2550

 dense_4 (Dense)         (None, 10)           510

 dense_5 (Dense)         (None, 10)           110

=================================================================
Total params: 96,820
Trainable params: 96,820
Non-trainable params: 0
_________________________________________________________________
```

Step 3: Apply refinement training to the low-rank model.
For refinement training the model was trained on the dataset for 10 epochs and a trial and error learning rate of .15 and no compression. Here are the results:

Epoch 10/10
1/79 [..............................] - ETA: 0s - loss: 0.0758 - accuracy: 0.98433/79 [===========>.................] - ETA: 0s - loss: 0.0935 - accuracy: 0.97165/79 [======================>......] - ETA: 0s - loss: 0.0864 - accuracy: 0.97479/79 [==============================] - 0s 4ms/step - loss: 0.0856 - accuracy: 0.9753 - val_loss: 0.0721 - val_accuracy: 0.9798   79/79 [==============================] - 0s 1ms/step

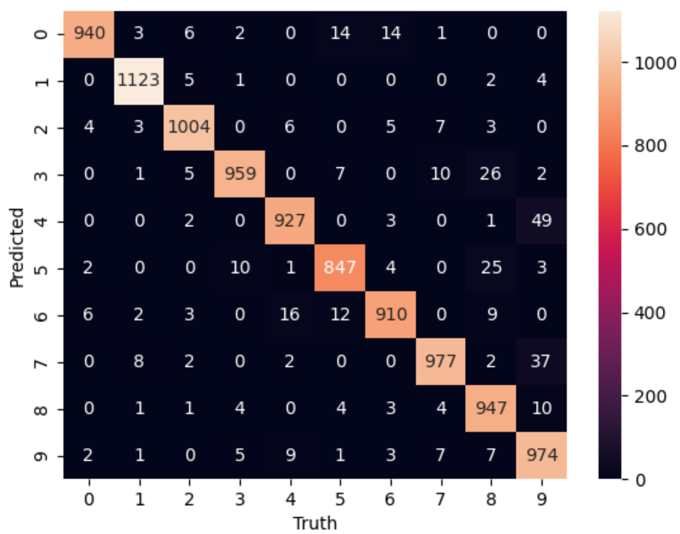




Step 4: Repeat steps 2 and 3 with 2x, 4x, and 8x compression. Report the same as the above results, and add in the model summary for each level of compression.

2x model compression: Learning rate: .1
The model performed close to the original with 2x compression.

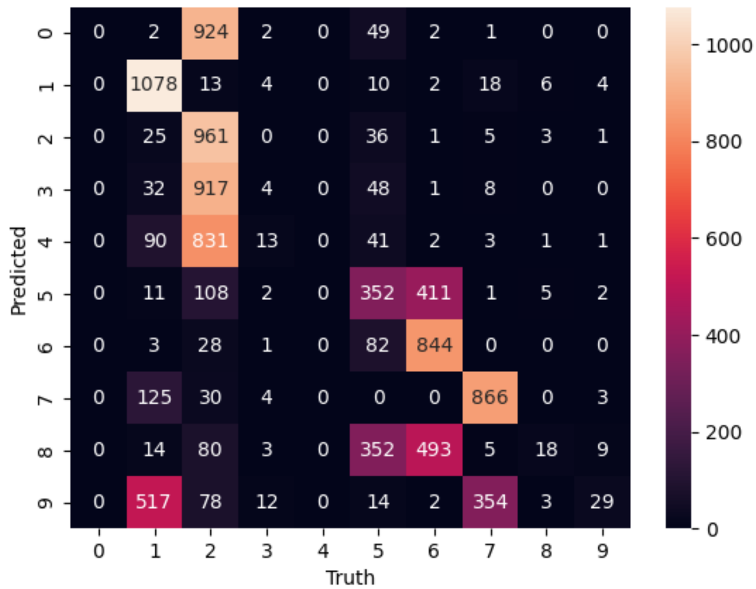
```
Epoch 10/10
 1/79 [..............................] - ETA: 0s - loss: 0.101828/79 [=========>..................] - ETA: 0s - loss: 0.160663/79 [=====================>.......] - ETA: 0s - loss: 0.150579/79 [==============================] - 0s 4ms/step - loss: 0.1477 - accuracy: 0.9606 - val_loss: 0.1330 - val_accuracy: 0.9608
79/79 [==============================] - 0s 1ms/step

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 50)                39250

 dense_1 (Dense)             (None, 100)               5100

 dense_2 (Dense)             (None, 25)                2525

 dense_3 (Dense)             (None, 50)                1300

 dense_4 (Dense)             (None, 5)                 255

 dense_5 (Dense)             (None, 10)                60

=================================================================
Total params: 48,490
Trainable params: 48,490
Non-trainable params: 0
_________________________________________________________________
```


Predicted
Truth

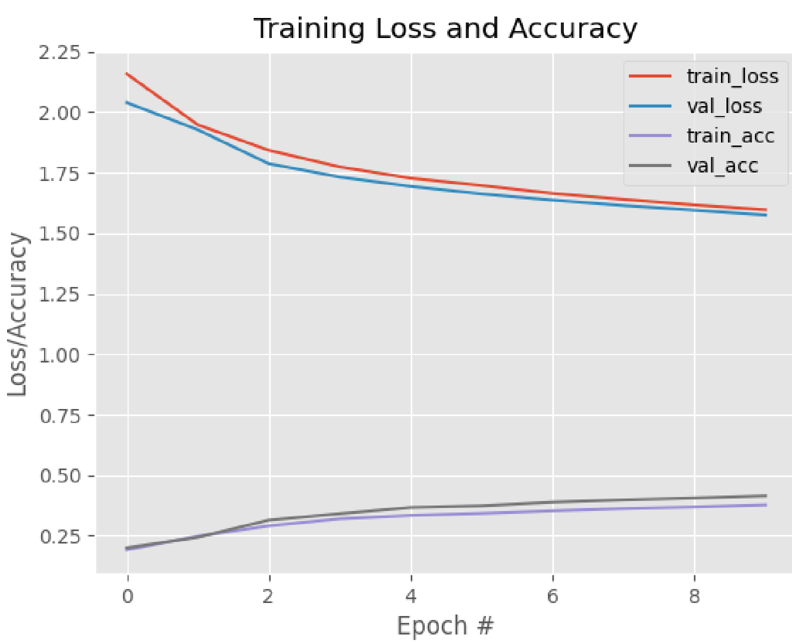

Training Loss and Accuracy
train_loss
val_loss
train_acc
val_acc
Loss/Accuracy
Epoch #

4x Compression: learning_rate = .035 – the model lost a lot of steam being compressed 4x. The diagonal pattern of the confusion matrix is somewhat preserved, but is starting to fall away.

```
Epoch 10/10
 1/79 [..............................] - ETA: 0s - loss: 1.6095 - accuracy: 0.337/79
[=============>................] - ETA: 0s - loss: 1.5998 - accuracy: 0.370/79
[=========================>....] - ETA: 0s - loss: 1.5948 - accuracy: 0.379/79
[==============================] - 0s 3ms/step - loss: 1.5968 - accuracy: 0.3778 - val_loss:
1.5753 - val_accuracy: 0.4152
79/79 [==============================] - 0s 1ms=>.....] - ETA: 0s - loss: 1.9773 - accuracy: 0

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 25)                19625

 dense_1 (Dense)             (None, 100)               2600

 dense_2 (Dense)             (None, 12)                1212

 dense_3 (Dense)             (None, 50)                650

 dense_4 (Dense)             (None, 2)                 102

 dense_5 (Dense)             (None, 10)                30

=================================================================
Total params: 24,219
Trainable params: 24,219
Non-trainable params: 0
```



Predicted
Truth

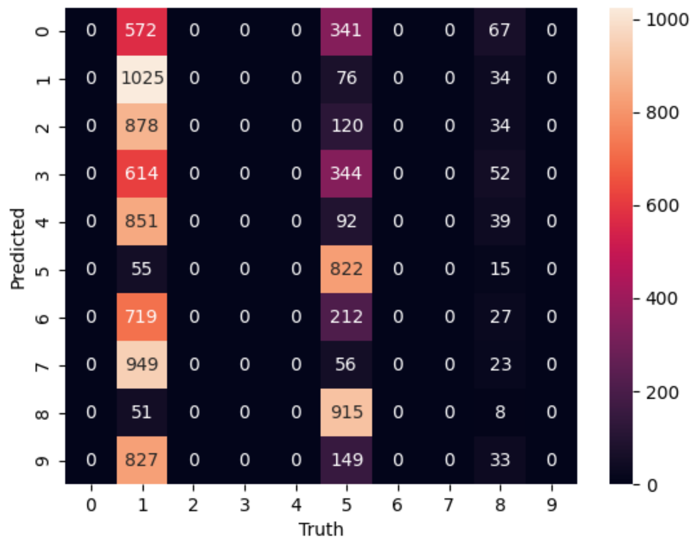


Training Loss and Accuracy
train_loss
val_loss
train_acc
val_acc
Loss/Accuracy
Epoch #

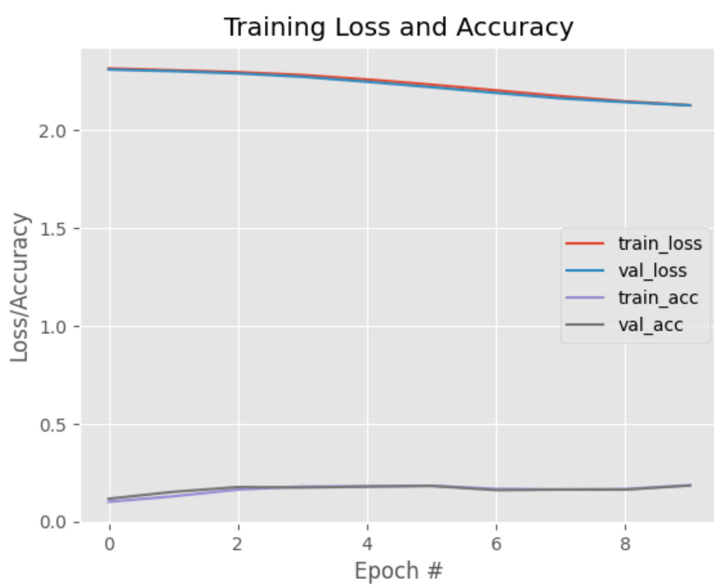
8x Compression: Learning rate: .015 – the model has lost most of its accuracy coming in at 18%. Tell tale signs of compression can be seen in the confusion matrix as the error is heightened.

```
Epoch 10/10
 1/79 [..............................] - ETA: 0s - loss: 2.1185 - accuracy: 0.32/79
[==========>..................] - ETA: 0s - loss: 2.1252 - accuracy: 0.69/79
[=======================>....] - ETA: 0s - loss: 2.1290 - accuracy: 0.79/79
[============================] - 0s 3ms/step - loss: 2.1290 - accuracy: 0.1874 - val_loss: 2.1283 - val_accuracy: 0.1855
79/79 [==============================] - 0s 1ms/step
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)            Output Shape            Param #
=================================================================
 flatten (Flatten)        (None, 784)            0

 dense (Dense)            (None, 12)             9420

 dense_1 (Dense)          (None, 100)            1300

 dense_2 (Dense)          (None, 6)              606

 dense_3 (Dense)          (None, 50)             350

 dense_4 (Dense)          (None, 1)              51

 dense_5 (Dense)          (None, 10)             20

=================================================================
Total params: 11,747
Trainable params: 11,747
Non-trainable params: 0
_________________________________________________________________
```



Predicted
Truth
0 572 0 0 0 341 0 0 67 0
0 1025 0 0 0 76 0 0 34 0
0 878 0 0 0 120 0 0 34 0
0 614 0 0 0 344 0 0 52 0
0 851 0 0 0 92 0 0 39 0
0 55 0 0 0 822 0 0 15 0
0 719 0 0 0 212 0 0 27 0
0 949 0 0 0 56 0 0 23 0
0 51 0 0 0 915 0 0 8 0
0 827 0 0 0 149 0 0 33 0
1000
800
600
400
200
0



Training Loss and Accuracy
Loss/Accuracy
Epoch #
train_loss
val_loss
train_acc
val_acc