Q1.

Model Name	Function
task1_lenet_withaug_adam.h5	Task1 with optimizer as Adam and augmentation
task1_lenet_withoutaug_adam.h5	Task 1 without augmentation
task1_lenet_withaug_sgd.h5	Task1 with optimizer as SGDand augmentation

As the Imageset was very much close to the Mnist Dataset in appearance thus I used a Lenet-5. Alexnet was seemingly too big to learn as it requires a big dataset with good quality images and Vggnet gave an error while an attempt.

Preprocessing:

1. Firstly I stacked pictures into a numpy array and set their corresponding labels as well.
2. The images were resized to 128*128*3 because the original size was too big to be processed and it was throwing out of bounds error for RAM
3. The class wise image set was of just 40 images which was pretty less for a decent accuracy thus to enlarge the database I have used 4 data augmentation techniques:
    a. Gaussian Noise
    b. Image Flip/Rotation
    c. Brightness
    d. Contrast

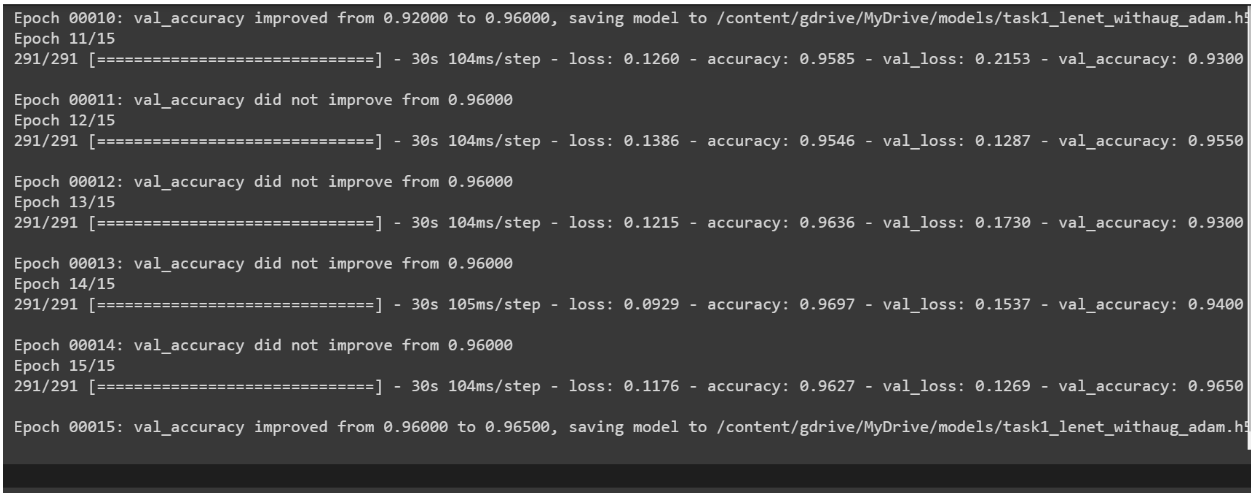
   By applying these techniques I have reproduced my dataset 5 times the original dataset preventing overfitting as well for the images and reflection of changes could be shown by table below:

Dataset	Minimum Loss	Maximum Val Accuracy	Maximum Test Accuracy
With Augmentation	0.09	96.5	98.3
Without Augmentation	0.6	50.00	47.06

Val Accuracy
Proven that with augmentation there are better results thus testing upon optimiser.

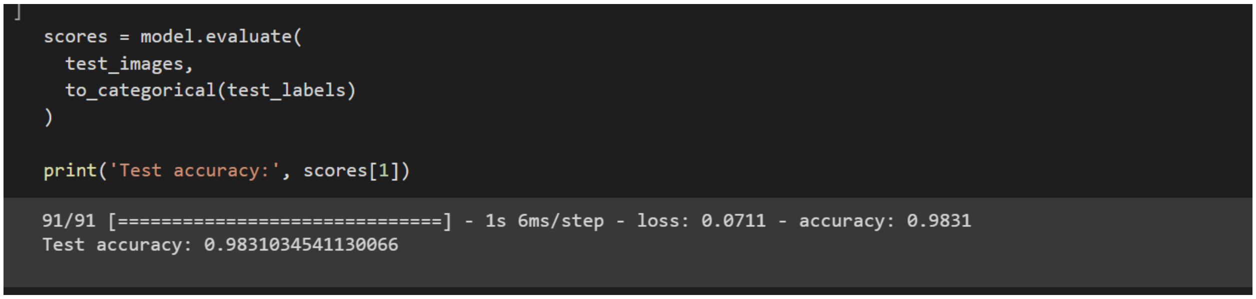
Optimiser	Minimum Loss	Maximum Val Accuracy	Maximum Test Accuracy
Adam	0.09	96.5	98.3
SGD	0.14	97.0	94.04

Adam Train accuracy



```
Epoch 00010: val_accuracy improved from 0.92000 to 0.96000, saving model to /content/gdrive/MyDrive/models/task1_lenet_withaug_adam.h5
Epoch 11/15
291/291 [==============================] - 30s 104ms/step - loss: 0.1260 - accuracy: 0.9585 - val_loss: 0.2153 - val_accuracy: 0.9300

Epoch 00011: val_accuracy did not improve from 0.96000
Epoch 12/15
291/291 [==============================] - 30s 104ms/step - loss: 0.1386 - accuracy: 0.9546 - val_loss: 0.1287 - val_accuracy: 0.9550

Epoch 00012: val_accuracy did not improve from 0.96000
Epoch 13/15
291/291 [==============================] - 30s 104ms/step - loss: 0.1215 - accuracy: 0.9636 - val_loss: 0.1730 - val_accuracy: 0.9300

Epoch 00013: val_accuracy did not improve from 0.96000
Epoch 14/15
291/291 [==============================] - 30s 105ms/step - loss: 0.0929 - accuracy: 0.9697 - val_loss: 0.1537 - val_accuracy: 0.9400

Epoch 00014: val_accuracy did not improve from 0.96000
Epoch 15/15
291/291 [==============================] - 30s 104ms/step - loss: 0.1176 - accuracy: 0.9627 - val_loss: 0.1269 - val_accuracy: 0.9650

Epoch 00015: val_accuracy improved from 0.96000 to 0.96500, saving model to /content/gdrive/MyDrive/models/task1_lenet_withaug_adam.h5
```

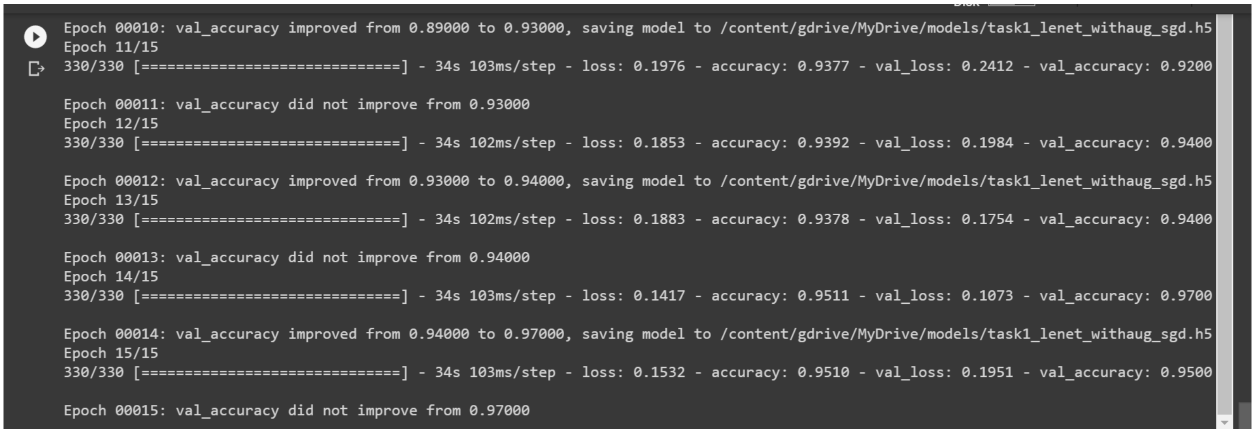
Adam Test Accuracy



```
scores = model.evaluate(
  test_images,
  to_categorical(test_labels)
)

print('Test accuracy:', scores[1])
```

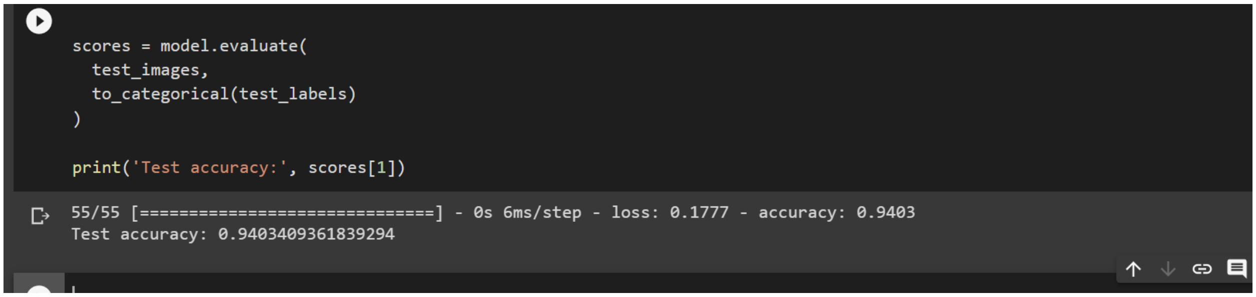
```
91/91 [==============================] - 1s 6ms/step - loss: 0.0711 - accuracy: 0.9831
Test accuracy: 0.9831034541130066
```

SGD Train accuracy



```
Epoch 00010: val_accuracy improved from 0.89000 to 0.93000, saving model to /content/gdrive/MyDrive/models/task1_lenet_withaug_sgd.h5
Epoch 11/15
330/330 [==============================] - 34s 103ms/step - loss: 0.1976 - accuracy: 0.9377 - val_loss: 0.2412 - val_accuracy: 0.9200

Epoch 00011: val_accuracy did not improve from 0.93000
Epoch 12/15
330/330 [==============================] - 34s 102ms/step - loss: 0.1853 - accuracy: 0.9392 - val_loss: 0.1984 - val_accuracy: 0.9400

Epoch 00012: val_accuracy improved from 0.93000 to 0.94000, saving model to /content/gdrive/MyDrive/models/task1_lenet_withaug_sgd.h5
Epoch 13/15
330/330 [==============================] - 34s 102ms/step - loss: 0.1883 - accuracy: 0.9378 - val_loss: 0.1754 - val_accuracy: 0.9400

Epoch 00013: val_accuracy did not improve from 0.94000
Epoch 14/15
330/330 [==============================] - 34s 103ms/step - loss: 0.1417 - accuracy: 0.9511 - val_loss: 0.1073 - val_accuracy: 0.9700

Epoch 00014: val_accuracy improved from 0.94000 to 0.97000, saving model to /content/gdrive/MyDrive/models/task1_lenet_withaug_sgd.h5
Epoch 15/15
330/330 [==============================] - 34s 103ms/step - loss: 0.1532 - accuracy: 0.9510 - val_loss: 0.1951 - val_accuracy: 0.9500

Epoch 00015: val_accuracy did not improve from 0.97000
```

SGD Test Accuracy



```
scores = model.evaluate(
  test_images,
  to_categorical(test_labels)
)

print('Test accuracy:', scores[1])
```

```
55/55 [==============================] - 0s 6ms/step - loss: 0.1777 - accuracy: 0.9403
Test accuracy: 0.9403409361839294
```

Q2.

Model Name	Function
pretrain_gray.h5	Grayscale pretrain model
pretrain_mnist_gray.h5	Grayscale Mnist model
pretrained_3_mnist.h5	RGB Mnist on Lenet randomly initialized
task2_pretrained_mnist_3.h5	Pretrained RGB Mnist
pretrained.h5	The pretrained model on task 1 given dataset

As the choice of networks is Lenet-5 which is proven mathematically and theoretically that is one of the best Convolutional Neural Network for the Mnist dataset. AlexNet is far deep network than required for this task and VggNet was also not found suitable because the Keras implementation requires at least 32*32*1 type images.

The dataset given for task 1 is of 1200*900*3 i.e. its color components are splitted in RGB whereas the Mnist is a Monotone database i.e. grayscale format thus to use the previous task network as a pretrained network we require inter conversion.

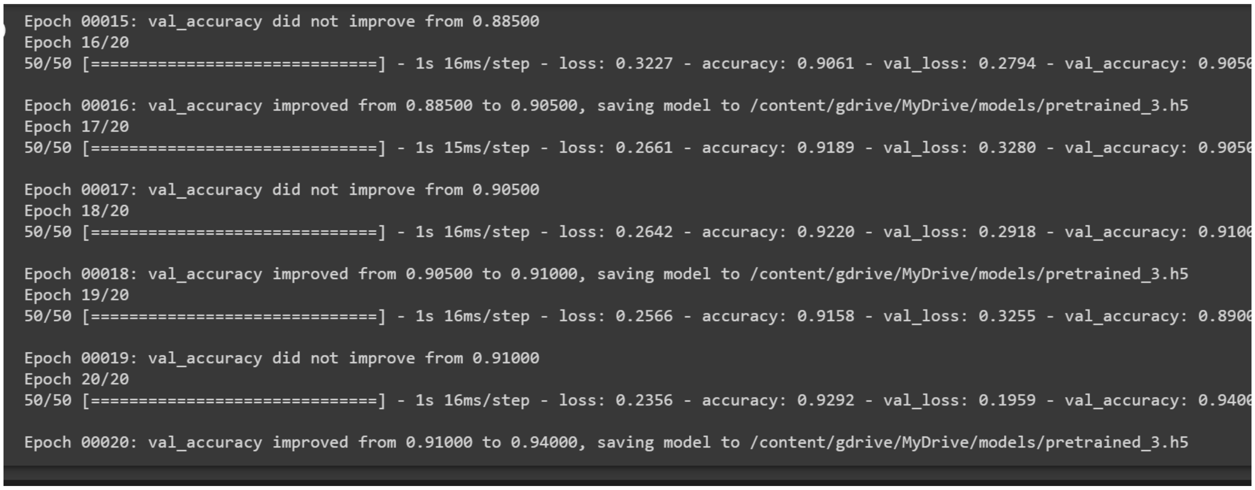
1. Mnist to RGB- The conversion was not really Grayscale to RGB conversion instead I stacked the same Mnist grayscale gradient thrice taking the axis about third dimension.
2. RGB dataset to grayscale using opencv

For the analysis I am referring to the first conversion but I have trained models with the second interconversions as well and you can find them in the model folder mentioned above.

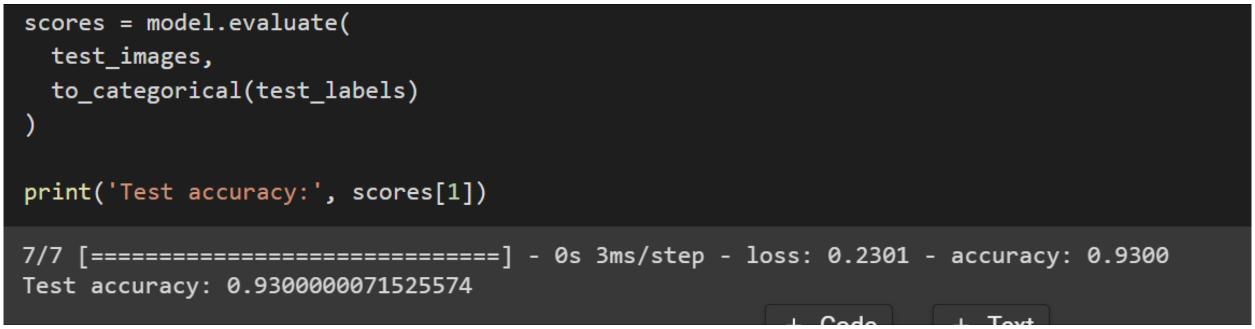
The details of pretrained models are as follows:

1) The model was trained of given images for 0-9 which accounts for images in total and dataset was increased by techniques of data augmentation such as gaussian noise,brightness change,contrast switch and rotation

2) Out of 2000 ,1700 were used for training images, 300 for testing
3) The model was trained on same Lenet with same hyperparameters as for the Mnist
4) The model was trained for 20 epochs due to less number of data and shuffle=true
5) The Val Accuracy was 94%
6) The model name is “pretrained_3.h5”



```
Epoch 00015: val_accuracy did not improve from 0.88500
Epoch 16/20
50/50 [==============================] - 1s 16ms/step - loss: 0.3227 - accuracy: 0.9061 - val_loss: 0.2794 - val_accuracy: 0.9050

Epoch 00016: val_accuracy improved from 0.88500 to 0.90500, saving model to /content/gdrive/MyDrive/models/pretrained_3.h5
Epoch 17/20
50/50 [==============================] - 1s 15ms/step - loss: 0.2661 - accuracy: 0.9189 - val_loss: 0.3280 - val_accuracy: 0.9050

Epoch 00017: val_accuracy did not improve from 0.90500
Epoch 18/20
50/50 [==============================] - 1s 16ms/step - loss: 0.2642 - accuracy: 0.9220 - val_loss: 0.2918 - val_accuracy: 0.9100

Epoch 00018: val_accuracy improved from 0.90500 to 0.91000, saving model to /content/gdrive/MyDrive/models/pretrained_3.h5
Epoch 19/20
50/50 [==============================] - 1s 16ms/step - loss: 0.2566 - accuracy: 0.9158 - val_loss: 0.3255 - val_accuracy: 0.8900

Epoch 00019: val_accuracy did not improve from 0.91000
Epoch 20/20
50/50 [==============================] - 1s 16ms/step - loss: 0.2356 - accuracy: 0.9292 - val_loss: 0.1959 - val_accuracy: 0.9400

Epoch 00020: val_accuracy improved from 0.91000 to 0.94000, saving model to /content/gdrive/MyDrive/models/pretrained_3.h5
```



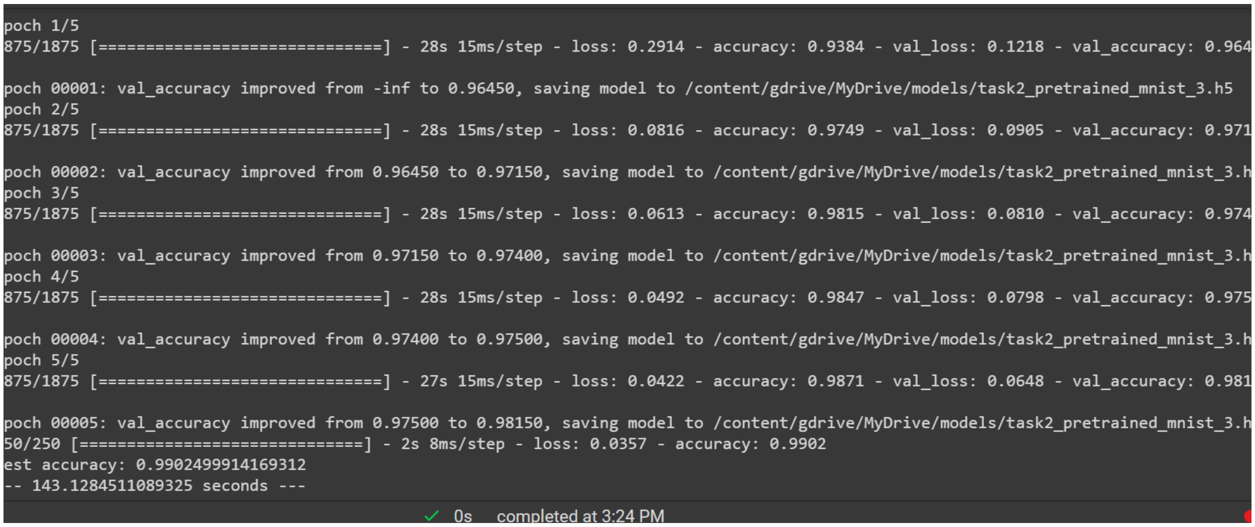
```python
scores = model.evaluate(
  test_images,
  to_categorical(test_labels)
)

print('Test accuracy:', scores[1])
```

```
7/7 [==============================] - 0s 3ms/step - loss: 0.2301 - accuracy: 0.9300
Test accuracy: 0.9300000071525574
```

The preprocessing was as follows:

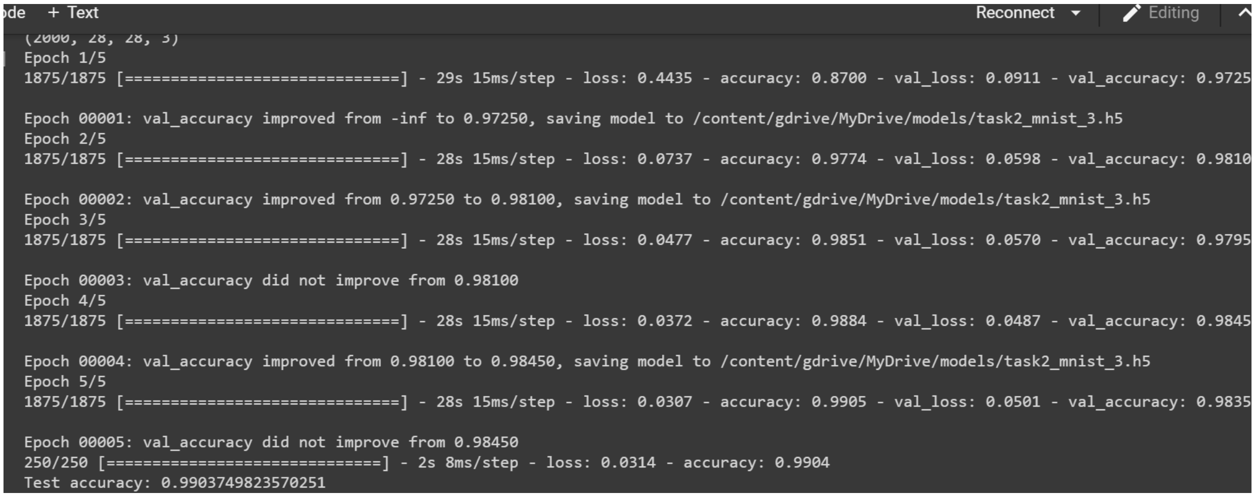
1) The validation set out of test set was kept 0f 2000 images for both the networks
2) Both the networks gave results upon the Lenet training however the second image depicts the stats of pretrained network whose specifications are mentioned in the first task.

The pretrained network



```
poch 1/5
875/1875 [==============================] - 28s 15ms/step - loss: 0.2914 - accuracy: 0.9384 - val_loss: 0.1218 - val_accuracy: 0.964

poch 00001: val_accuracy improved from -inf to 0.96450, saving model to /content/gdrive/MyDrive/models/task2_pretrained_mnist_3.h5
poch 2/5
875/1875 [==============================] - 28s 15ms/step - loss: 0.0816 - accuracy: 0.9749 - val_loss: 0.0905 - val_accuracy: 0.971

poch 00002: val_accuracy improved from 0.96450 to 0.97150, saving model to /content/gdrive/MyDrive/models/task2_pretrained_mnist_3.h
poch 3/5
875/1875 [==============================] - 28s 15ms/step - loss: 0.0613 - accuracy: 0.9815 - val_loss: 0.0810 - val_accuracy: 0.974

poch 00003: val_accuracy improved from 0.97150 to 0.97400, saving model to /content/gdrive/MyDrive/models/task2_pretrained_mnist_3.h
poch 4/5
875/1875 [==============================] - 28s 15ms/step - loss: 0.0492 - accuracy: 0.9847 - val_loss: 0.0798 - val_accuracy: 0.975

poch 00004: val_accuracy improved from 0.97400 to 0.97500, saving model to /content/gdrive/MyDrive/models/task2_pretrained_mnist_3.h
poch 5/5
875/1875 [==============================] - 27s 15ms/step - loss: 0.0422 - accuracy: 0.9871 - val_loss: 0.0648 - val_accuracy: 0.981

poch 00005: val_accuracy improved from 0.97500 to 0.98150, saving model to /content/gdrive/MyDrive/models/task2_pretrained_mnist_3.h
50/250 [==============================] - 2s 8ms/step - loss: 0.0357 - accuracy: 0.9902
est accuracy: 0.9902499914169312
-- 143.1284511089325 seconds ---
```

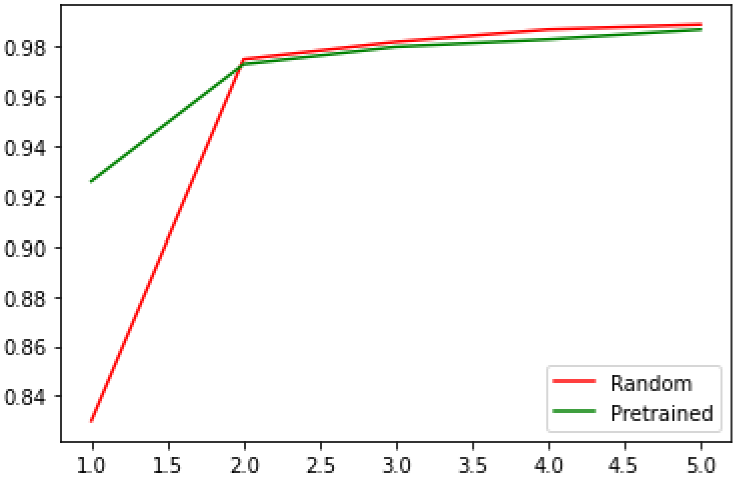
The randomly initialized



```
(2000, 28, 28, 3)
Epoch 1/5
1875/1875 [==============================] - 29s 15ms/step - loss: 0.4435 - accuracy: 0.8700 - val_loss: 0.0911 - val_accuracy: 0.9725

Epoch 00001: val_accuracy improved from -inf to 0.97250, saving model to /content/gdrive/MyDrive/models/task2_mnist_3.h5
Epoch 2/5
1875/1875 [==============================] - 28s 15ms/step - loss: 0.0737 - accuracy: 0.9774 - val_loss: 0.0598 - val_accuracy: 0.9810

Epoch 00002: val_accuracy improved from 0.97250 to 0.98100, saving model to /content/gdrive/MyDrive/models/task2_mnist_3.h5
Epoch 3/5
1875/1875 [==============================] - 28s 15ms/step - loss: 0.0477 - accuracy: 0.9851 - val_loss: 0.0570 - val_accuracy: 0.9795

Epoch 00003: val_accuracy did not improve from 0.98100
Epoch 4/5
1875/1875 [==============================] - 28s 15ms/step - loss: 0.0372 - accuracy: 0.9884 - val_loss: 0.0487 - val_accuracy: 0.9845

Epoch 00004: val_accuracy improved from 0.98100 to 0.98450, saving model to /content/gdrive/MyDrive/models/task2_mnist_3.h5
Epoch 5/5
1875/1875 [==============================] - 28s 15ms/step - loss: 0.0307 - accuracy: 0.9905 - val_loss: 0.0501 - val_accuracy: 0.9835

Epoch 00005: val_accuracy did not improve from 0.98450
250/250 [==============================] - 2s 8ms/step - loss: 0.0314 - accuracy: 0.9904
Test accuracy: 0.9903749823570251
```

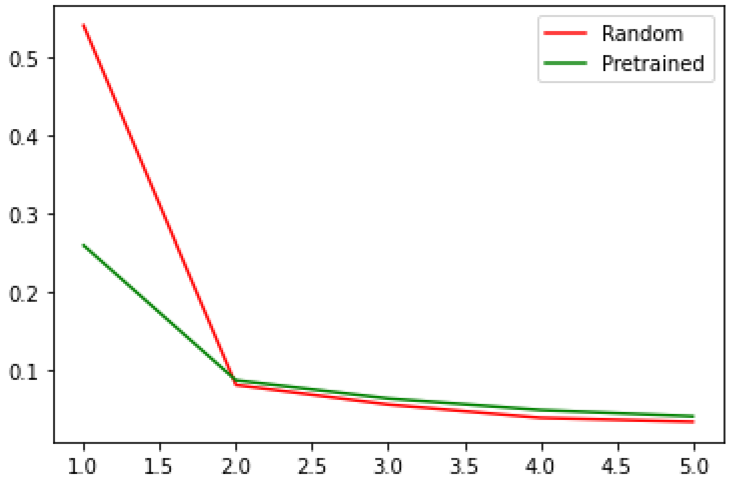
The first image is of Lenet-Mnist and second image depicts the Lenet-Mnist with a pretrained model and analysis was as follows

Accuracy



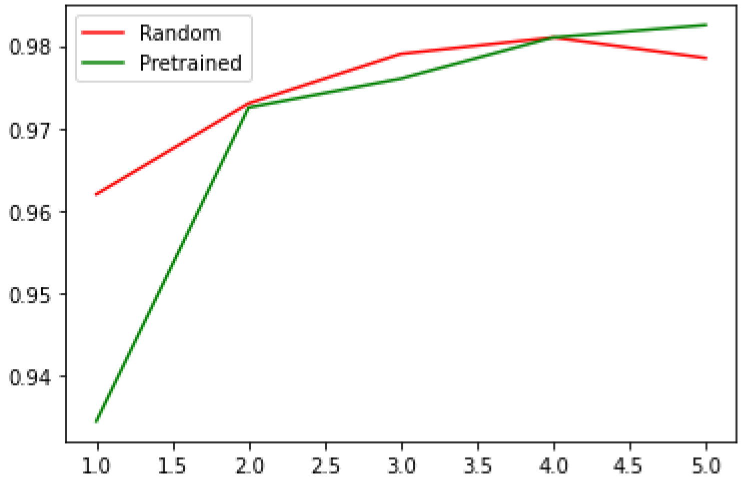
The accuracy is similar just that the random initialized network at first epoch is a bit random and uncertain thus starts from away but the graph slope is somewhat parallel.

Loss



The loss is similar just that the random initialized network at first epoch is a bit random and uncertain thus starts from away but the graph slope is somewhat parallel.

Validation Accuracy



The validation accuracy of randomly initialized network is a bit uneven because there is no prior belief and every feature has to be learnt from scratch thus is very much dependent on the training set of images whereas there is no such great deviation in pretrained network and it tends to converge at its maxima

Conclusion

As Lenet is already hyperparamterised theoretically and practically to give best results on Mnist thus performs very much as good as the pretrained network but for the initial epochs and accuracy best results are given with pre trained network.

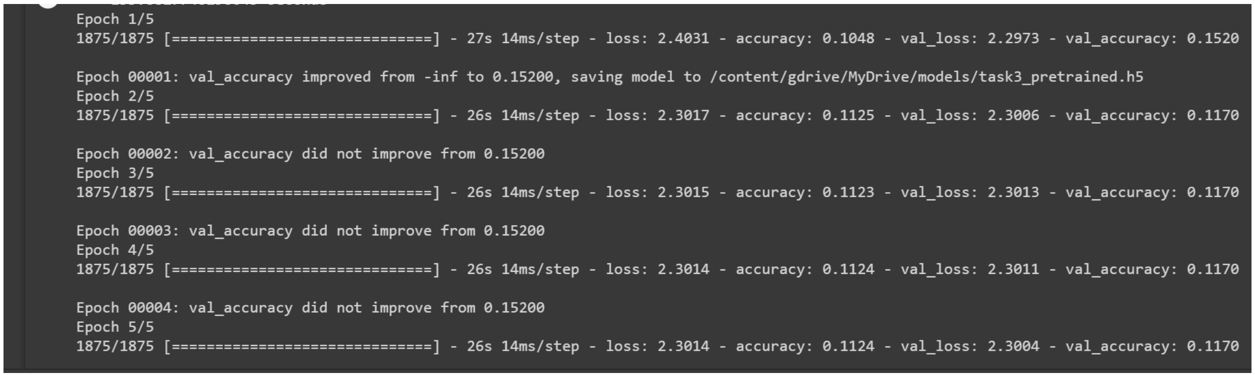
Q3

Model	Function
task3_pretrained.h5	The best final network upon pretrained network
task3_random.h5	The best final network upon randomly initialized network
pretrained_2.h5	The network which is pretrained from the dataset of first tasks

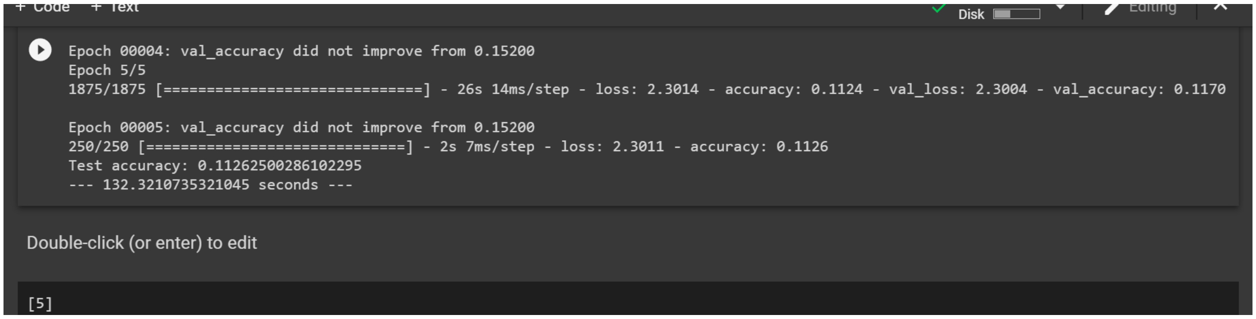
Model Type	Maximum Loss	Max Val Accuracy	Min Accuracy	Min Val Loss
Pretrained	2.4031	15.20	10.48	2.297
Random	2.3023	11.70	10.78	2.3006

As the first network had a certain prior belief thus the maximum loss was much more than any other epoch and the max loss occurred at the first epoch itself.
As the weights had not been changed drastically post the first epoch due to certain prior belief in the network thus during the first epoch the pretrained network performed much better. With the similar belief Validation loss was also less until the network weights were polluted by such weights again thus leading to similar situation until the first epoch

Pretrained



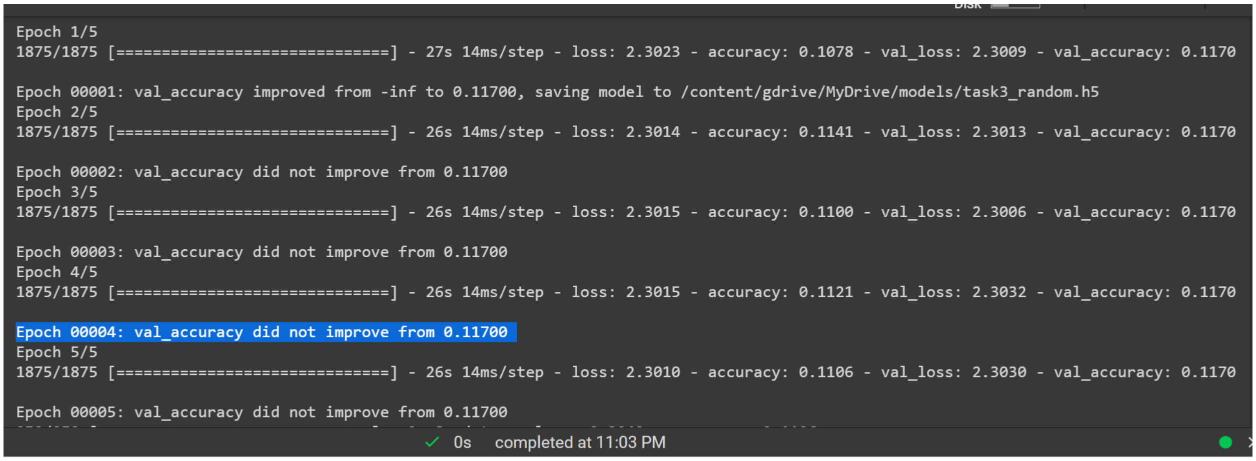
```
Epoch 1/5
1875/1875 [==============================] - 27s 14ms/step - loss: 2.4031 - accuracy: 0.1048 - val_loss: 2.2973 - val_accuracy: 0.1520

Epoch 00001: val_accuracy improved from -inf to 0.15200, saving model to /content/gdrive/MyDrive/models/task3_pretrained.h5
Epoch 2/5
1875/1875 [==============================] - 26s 14ms/step - loss: 2.3017 - accuracy: 0.1125 - val_loss: 2.3006 - val_accuracy: 0.1170

Epoch 00002: val_accuracy did not improve from 0.15200
Epoch 3/5
1875/1875 [==============================] - 26s 14ms/step - loss: 2.3015 - accuracy: 0.1123 - val_loss: 2.3013 - val_accuracy: 0.1170

Epoch 00003: val_accuracy did not improve from 0.15200
Epoch 4/5
1875/1875 [==============================] - 26s 14ms/step - loss: 2.3014 - accuracy: 0.1124 - val_loss: 2.3011 - val_accuracy: 0.1170

Epoch 00004: val_accuracy did not improve from 0.15200
Epoch 5/5
1875/1875 [==============================] - 26s 14ms/step - loss: 2.3014 - accuracy: 0.1124 - val_loss: 2.3004 - val_accuracy: 0.1170
```



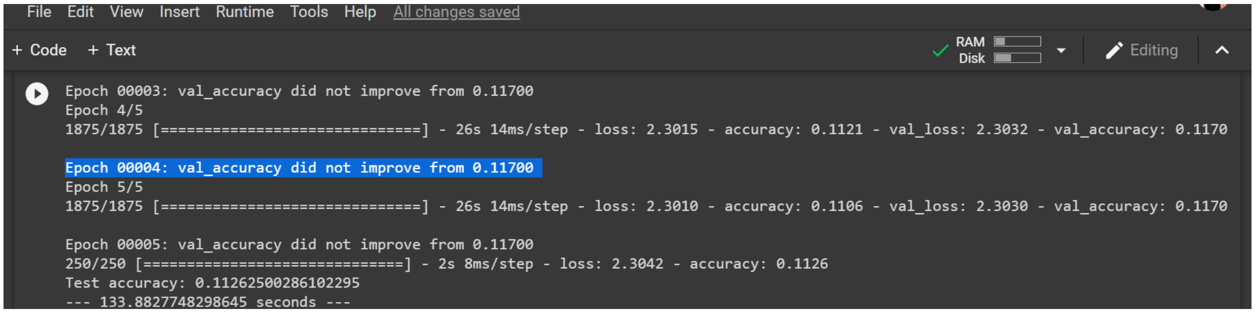
```
Epoch 00004: val_accuracy did not improve from 0.15200
Epoch 5/5
1875/1875 [==============================] - 26s 14ms/step - loss: 2.3014 - accuracy: 0.1124 - val_loss: 2.3004 - val_accuracy: 0.1170

Epoch 00005: val_accuracy did not improve from 0.15200
250/250 [==============================] - 2s 7ms/step - loss: 2.3011 - accuracy: 0.1126
Test accuracy: 0.11262500286102295
--- 132.3210735321045 seconds ---
```

Double-click (or enter) to edit

Random



```
Epoch 1/5
1875/1875 [==============================] - 27s 14ms/step - loss: 2.3023 - accuracy: 0.1078 - val_loss: 2.3009 - val_accuracy: 0.1170

Epoch 00001: val_accuracy improved from -inf to 0.11700, saving model to /content/gdrive/MyDrive/models/task3_random.h5
Epoch 2/5
1875/1875 [==============================] - 26s 14ms/step - loss: 2.3014 - accuracy: 0.1141 - val_loss: 2.3013 - val_accuracy: 0.1170

Epoch 00002: val_accuracy did not improve from 0.11700
Epoch 3/5
1875/1875 [==============================] - 26s 14ms/step - loss: 2.3015 - accuracy: 0.1100 - val_loss: 2.3006 - val_accuracy: 0.1170

Epoch 00003: val_accuracy did not improve from 0.11700
Epoch 4/5
1875/1875 [==============================] - 26s 14ms/step - loss: 2.3015 - accuracy: 0.1121 - val_loss: 2.3032 - val_accuracy: 0.1170

Epoch 00004: val_accuracy did not improve from 0.11700
Epoch 5/5
1875/1875 [==============================] - 26s 14ms/step - loss: 2.3010 - accuracy: 0.1106 - val_loss: 2.3030 - val_accuracy: 0.1170

Epoch 00005: val_accuracy did not improve from 0.11700
```



```
Epoch 00003: val_accuracy did not improve from 0.11700
Epoch 4/5
1875/1875 [==============================] - 26s 14ms/step - loss: 2.3015 - accuracy: 0.1121 - val_loss: 2.3032 - val_accuracy: 0.1170

Epoch 00004: val_accuracy did not improve from 0.11700
Epoch 5/5
1875/1875 [==============================] - 26s 14ms/step - loss: 2.3010 - accuracy: 0.1106 - val_loss: 2.3030 - val_accuracy: 0.1170

Epoch 00005: val_accuracy did not improve from 0.11700
250/250 [==============================] - 2s 8ms/step - loss: 2.3042 - accuracy: 0.1126
Test accuracy: 0.11262500286102295
--- 133.8827748298645 seconds ---
```

References

For data augmentation techniques:
https://towardsdatascience.com/data-augmentation-compilation-with-python-and-opencv-a5a98a6906aa
For lenet code:
https://medium.com/@mgazar/lenet-5-in-9-lines-of-code-using-keras-ac99294c8086