Mohamed ABDUL GAFOOR

- We ran all the experiments in the **Google Colab** and the configuration is as follow. We have used the Google **GPU not CPU!**

CPU	GPU	TPU
Intel Xeon Processor with two cores @ 2.30 GHz and 13 GB RAM	Up to Tesla K80 with 12 GB of GDDR5 VRAM, Intel Xeon Processor with two cores @ 2.20 GHz and 13 GB RAM	Cloud TPU with 180 teraflops of computation, Intel Xeon Processor with two cores @ 2.30 GHz and 13 GB RAM

Exercise 1: In order to have a graphical description of the Fashion MNIST dataset display the first 9 images existent in the `trainX` variable using the OpenCV library.

Because **cv2.imshow()** is disabled in Colab, I could not use the openCV library for the visualization.

WARNING:
```
DisabledFunctionError: cv2.imshow() is disabled in Colab, because it causes
Jupyter sessions to crash; see https://github.com/jupyter/notebook/issues/3935.
```

Hence, it was visualized in the following way. This is from the Fashion MNIST data-set, trainX images.

```
%matplotlib inline
fig=plt.figure(figsize=(8, 8))
col = 3
row = 3
for i in range(1, col*row +1):
    img = trainX[i-1]
    fig.add_subplot(row, col, i)
    plt.imshow(img)
plt.show()
```

Alternatively, we can also visualize like below in the Colab Jupyter Notebook by importing **google.colab.patches import cv2_imshow**;

```
from google.colab.patches import cv2_imshow
for i in range(1, 10):
  img = trainX[i-1]
  cv2_imshow(img)
```

Exercise 2: Modify the number of filters in the convolutional layer as specified in Table 1. How is the system accuracy influenced by this parameter? How about the convergence time?

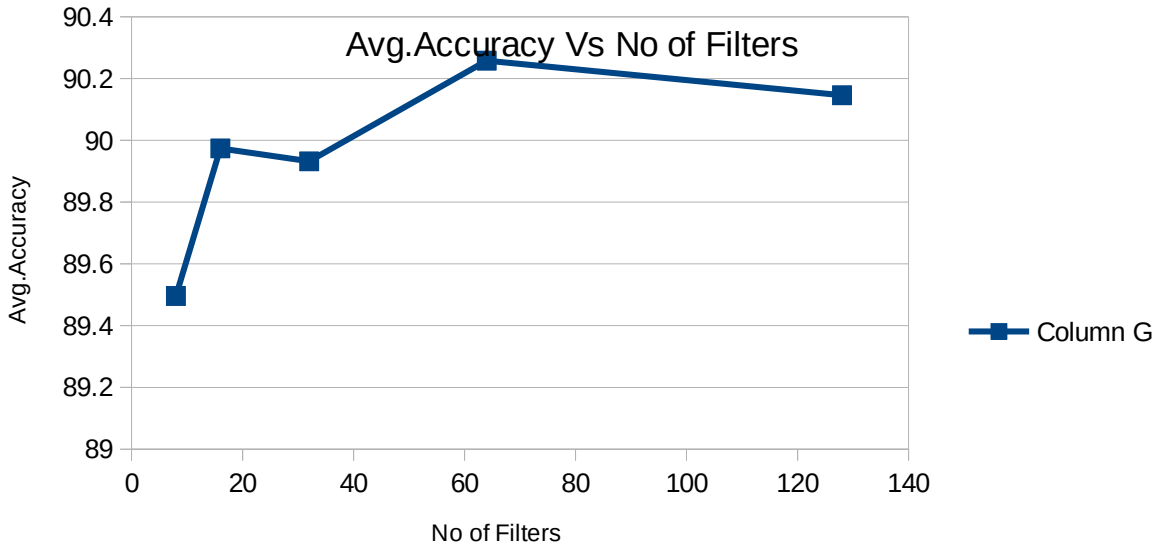
A screen shot from an excel sheet to show the average of 5 readings.

No of filters	Accuracy -1	Accuracy -2	Accuracy -3	Accuracy -4	Accuracy-5	Average Accuracy	time-1	Time-2	Time-3	Time-4	Time-5	Avg.Convergence time
8	89.91	88.85	89.68	89.39	89.65	89.496	0.00036	0.00038	0.00034	0.00012	0.0005	0.00034
16	89.75	90.13	90.09	90.2	89.7	89.974	0.00019	0.00398	2.00E-05	0.00014	0.00076	0.001018
32	89.78	89.6	90.84	89.72	89.72	89.932	0.00034	0.00053	0.00024	1.00E-05	0.00115	0.000454
64	89.99	90.65	90.17	89.97	90.51	90.258	3.00E-05	0.00017	9.00E-05	9.00E-05	0.00039	0.000154
128	90.52	90.2	90.09	89.87	90.05	90.146	0.00017	0.00029	0.00069	0.00026	0.00013	0.000308

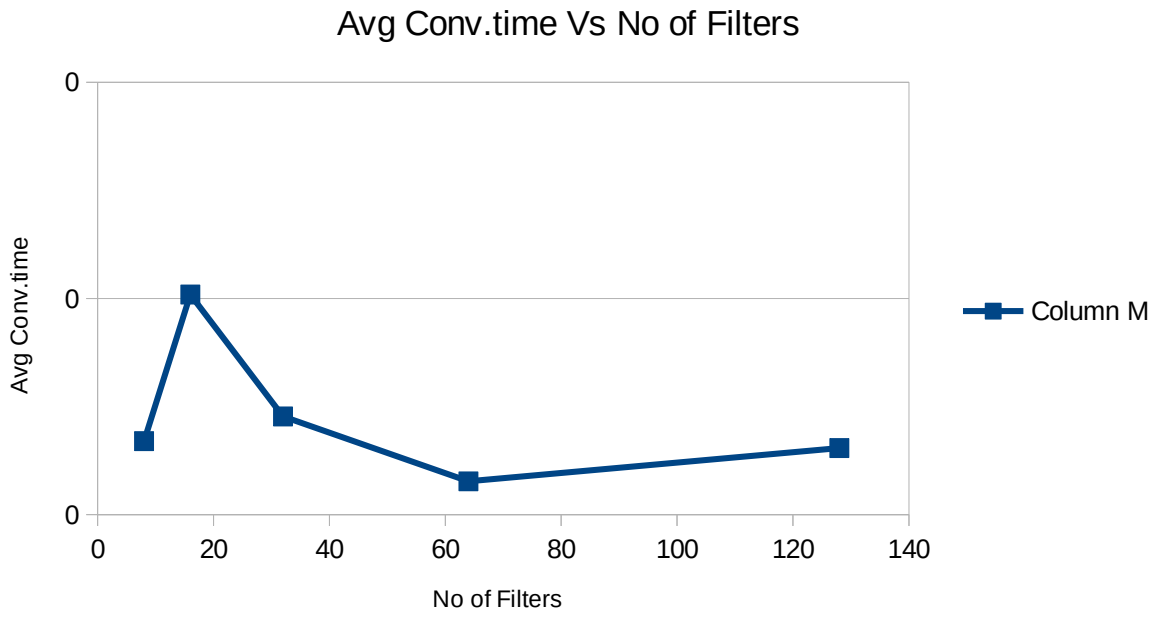
Mohamed ABDUL GAFOOR

The table and the following figure below show how the accuracy of the network increases with the number of neurons in the hidden layer.

No of filters	8	16	32	64	128
System accuracy	89.496	89.974	89.932	90.258	90.146
Convergence time	0.00034	0.001018	0.000454	1.54E-04	0.000308



The below graph shows convergence time with the number of filters. It is important to note that they model is not converging in 5 epochs, obviously it needs more epochs to converge.



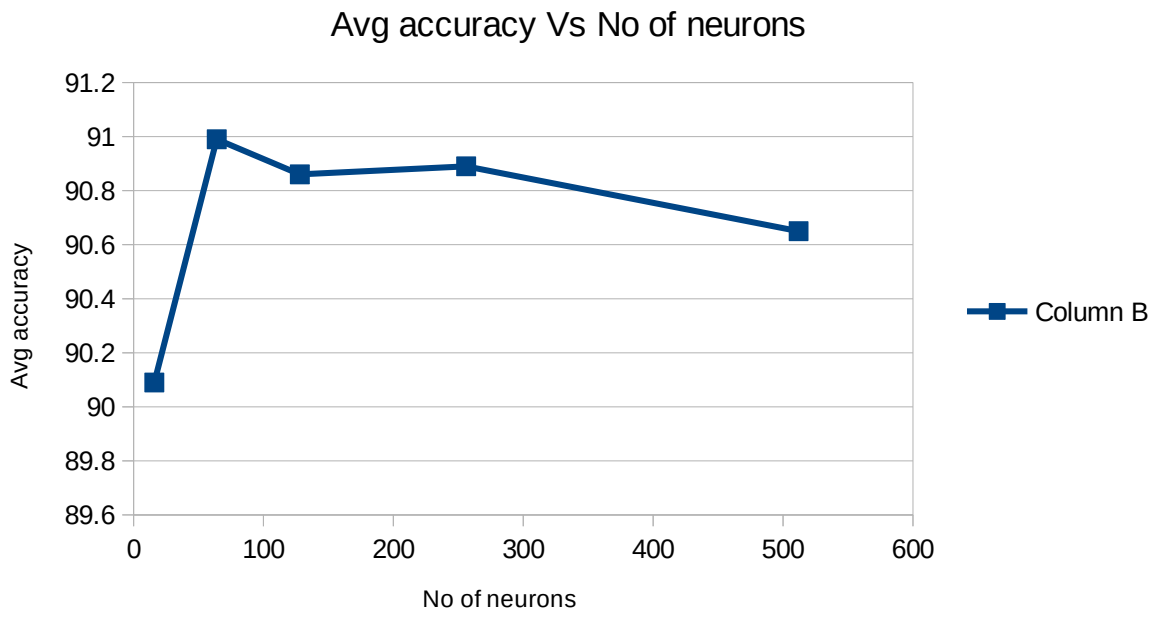
Mohamed ABDUL GAFOOR

Exercise 3: Modify the number of neurons in the dense hidden layer as specified in Table 2. The number of filters in the convolutional layer remains set to 32. What can be observed regarding the system accuracy?

The number of filters was set to 32. The following table shows the average number of reading.

No of filters	16	64	128	256	512
Avg. Accuracy	90.09	90.99	90.86	90.89	90.65

The below graph shows the no of filters and the accuracy.



Mohamed ABDUL GAFOOR

Exercise 4: Modify the number of epochs used to train the model as specified in Table 3. The number of neurons in the dense hidden layer is set to 16. What can be observed regarding the system accuracy? How about the convergence time?

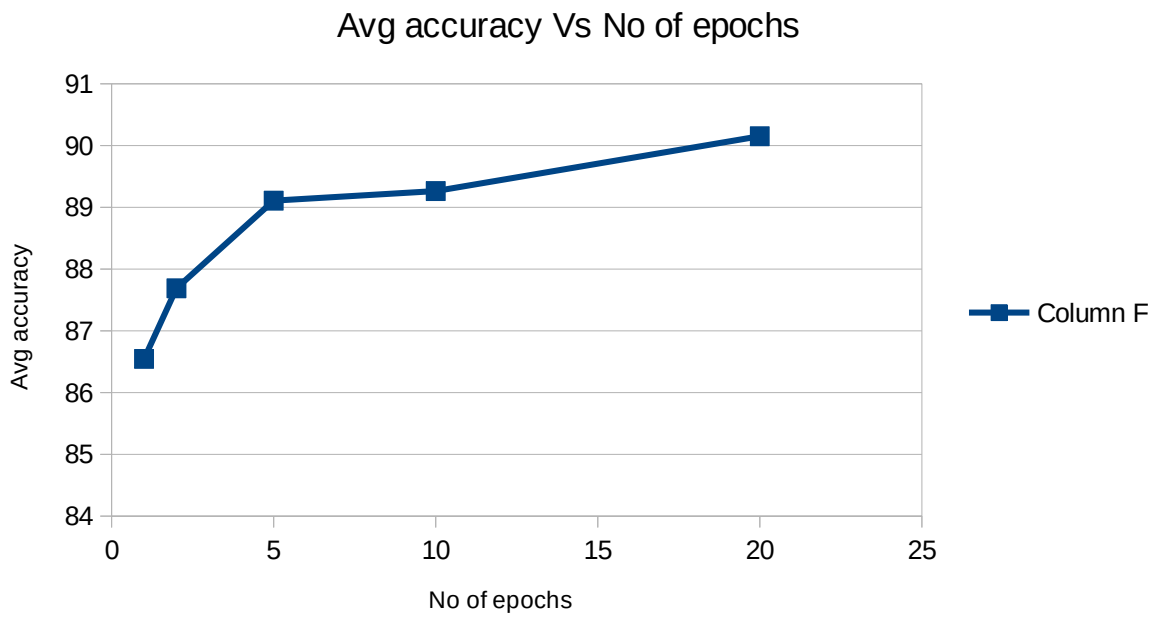
A screen shot from an excel sheet to show the average of 5 readings.

epochs	Accuracy1	Accuracy2	Accuracy3	Accuracy4	Avg accuracy	time1	time2	time3	time4	Avg time
1	85.6	85.96	86.59	88.03	86.545	0.00262	0.00067	0.00368	0.00151	0.00212
2	88.59	87.01	87.55	87.6	87.6875	0.00015	0.00043	0.00055	0.00145	0.000645
5	89.15	89.92	89.14	88.22	89.1075	7.00E-05	0.00179	5.00E-05	0.00022	0.000113333
10	88.74	88.57	89.86	89.88	89.2625	0.00063	0.00049	9.00E-05	0.00022	0.0003575
20	90.26	90.08	90.35	89.91	90.15	2.00E-05	0.00016	0.00042	0.00054	0.000285

The table below shows the average reading.

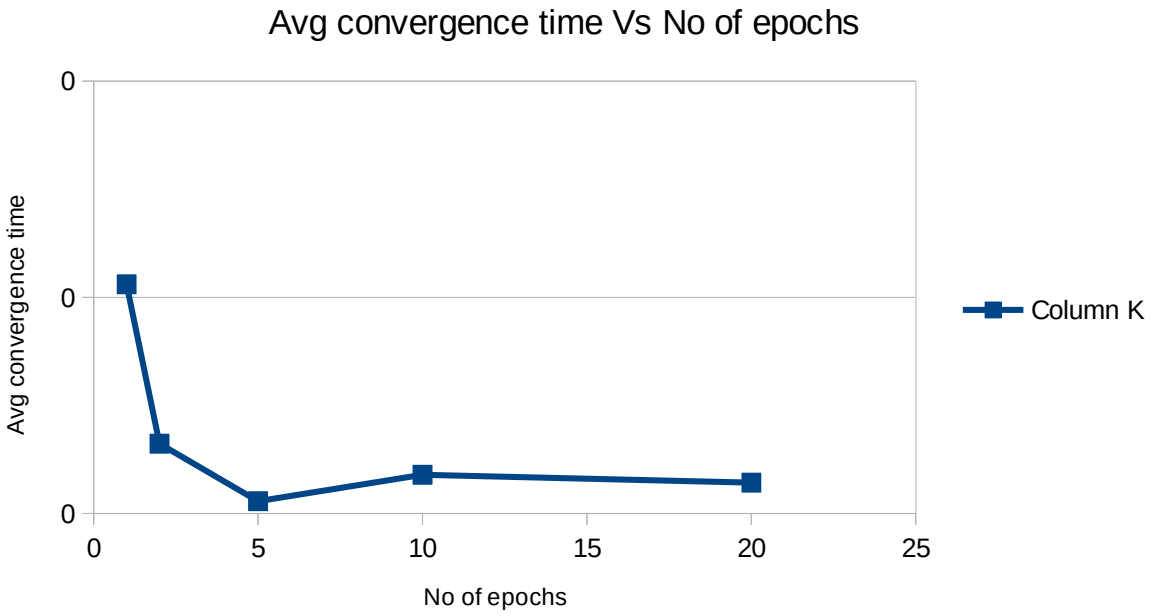
No of epochs	1	2	5	10	20
Avg accuracy	86.545	87.6875	89.1075	89.2625	90.15
Convergence time	0.00212	0.000645	0.0001133	0.0003575	0.000285

We can see if the number of epochs increases, the avg accuracy increases as well.



Mohamed ABDUL GAFOOR

The below figure shows the number of epochs Vs the avg convergence time. We can see the convergence time decreases with the epoch numbers. The model is not converging if the epoch size is 1, 2, 5, 10. If we set the epoch size to 20, the model converges around 12-13 epochs.



Exercise 5: Modify the learning rate of the stochastic gradient descend as presented in Table 4. Train the model for 5 epochs. What can be observed regarding the system accuracy and the convergence time?

A screen shot from an excel sheet to show the average of 4 readings.

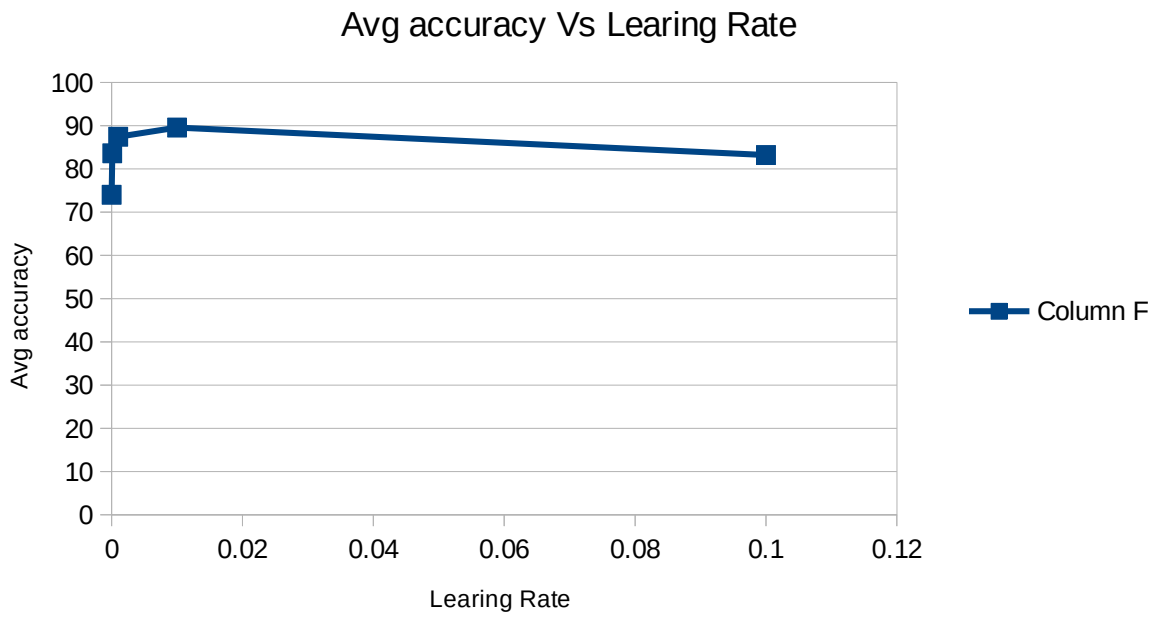
learingRate	Accuracy1	Accuracy2	Accuracy3	Accuracy4	avg acc	Convergence	time1	time2	time3	time4	Avg time
0.1	82.08	82.15	85.07	83.54	83.21	NO	0.00042	0.00139	0.00033	0.00073	0.0007175
0.01	88.59	89.91	89.95	89.76	89.5525	NO	0.00016	0.00046	0.00085	0.00051	0.000495
0.001	87.26	87.33	86.86	88.22	87.4175	NO	0.00176	0.00093	0.00238	0.00035	0.001355
0.0001	83.91	83.04	83.59	83.89	83.6075	NO	0.00041	0.00118	0.00229	0.00035	0.0010575
1E-05	74.55	73.19	74.78	73.56	74.02	NO	0.00033	2.00E-05	0.00249	0.00308	0.001863333

The table below shows the average reading.

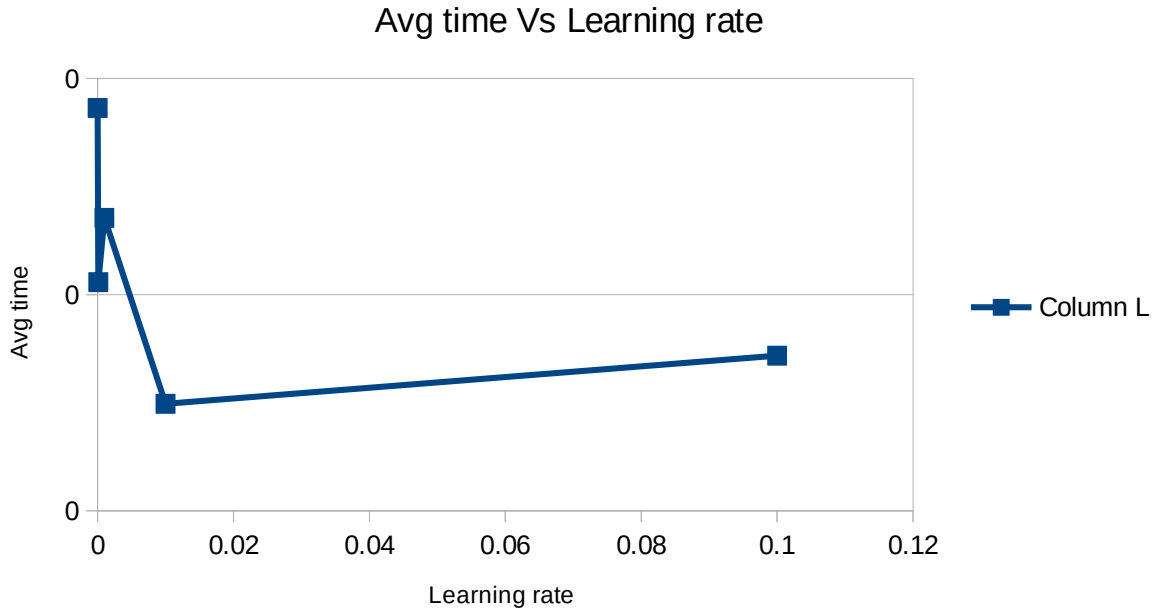
Learning rate	0.1	0.01	0.001	0.0001	0.00001
Avg accuracy	83.21	89.5525	87.4175	83.6075	74.02
Convergence time	0.0007175	0.000495	0.001355	0.0010575	0.0018633

Mohamed ABDUL GAFOOR

The below figure shows how the accuracy changes with respect to the learning rate. Accuracy is good if the learning rate is small, however, if it is too small it will degrade the model performance. For example in this case, if the learning rate is 0.00001, we can see the accuracy is going down.



Moreover, it does not converge for 5 epochs. May be we have to set more epoch size to observe the convergence. The below figure show the total time taken to fit the model for each learning rate.



Mohamed ABDUL GAFOOR

Exercise 6: Explore how adding regularization impacts model performance as compared to the baseline model. Add a **Dropout** layer on the CNN (after the **MaxPooling** layer) that randomly excludes 20% of neurons (`model.add(Dropout(0.2))`). Select for the learning rate a value of 0.01. Analyze the convergence speed with and without this regularization operation?

Without regularization, the accuracy is 89.39% and it converges in 0.00062. We set to 20 epochs, however, the system converges in 7 epochs.

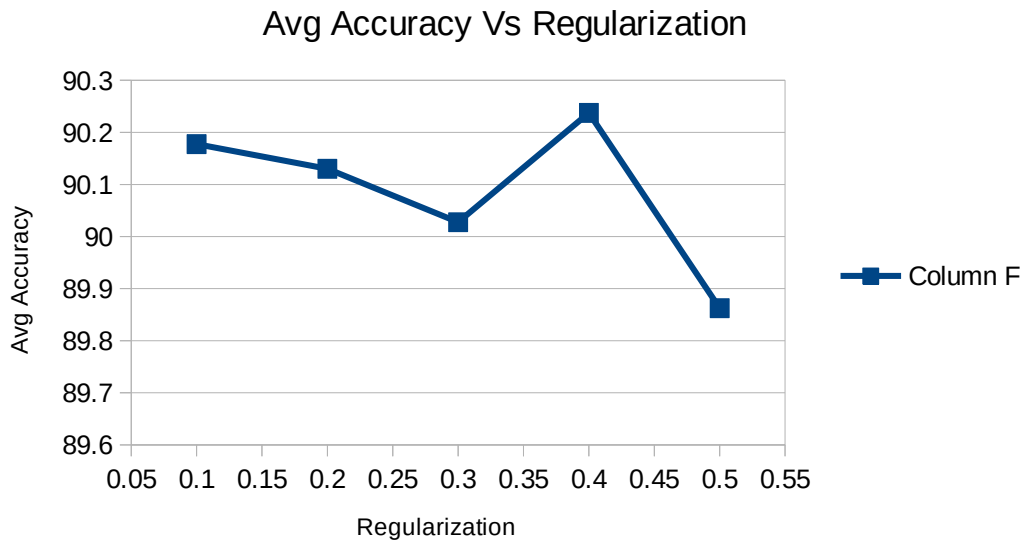
We set the patience = 3 in our callback function. A screen shot from an excel sheet to show the average of 4 readings.

Regularization	Accuracy1	Accuracy2	Accuracy3	Accuracy	Avg accu	time1	time2	time3	time4	Avg conv time
0.1	90.48	89.96	89.68	90.59	90.1775	0.00259	0.00043	0.00128	0.00072	0.001255
0.2	90.23	90.44	90.34	89.51	90.13	0.00081	5.00E-05	0.00026	0.00577	0.0017225
0.3	89.99	90.3	90.02	89.8	90.0275	0.00389	6.00E-05	0.00147	0.00245	0.0019675
0.4	90.34	89.76	90.92	89.93	90.2375	0.00306	0.00031	0.00361	0.00028	0.0012166667
0.5	89.84	89.94	89.55	90.12	89.8625	0.00099	0.00106	1.00E-05	0.0011	0.00079

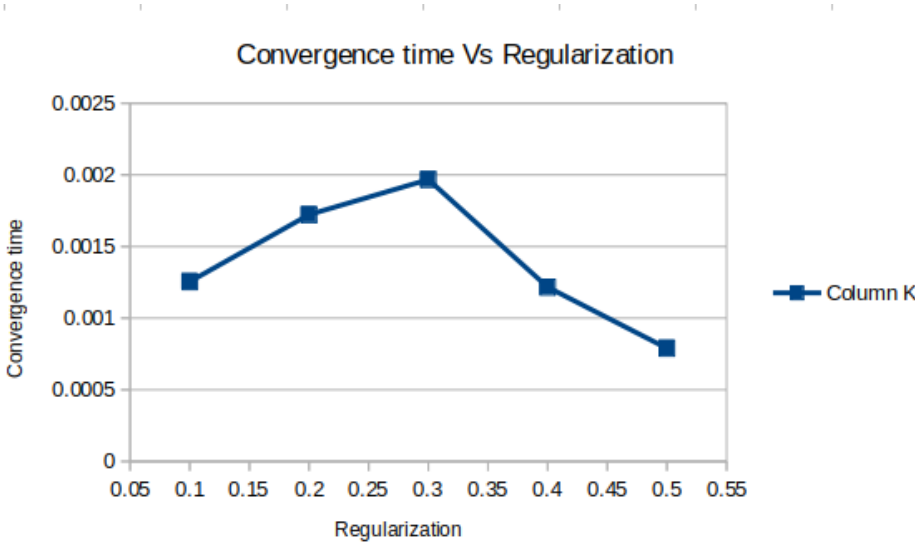
The table below shows the average reading.

Drop out %	0.1	0.2	0.3	0.4	0.5
Accuracy	90.1775	90.13	90.0275	90.2375	89.8625
Conv time	0.001255	0.0017225	0.0019675	0.00121	0.00079

The below figure shows the accuracy changes with the drop out regularization. Our average reading suggest that we get the best performance at 0.4.



The below is the convergence time. The convergence time is higher for the drop out of 0.4.



We can see that there is a slight improvement if we introduce regularization. Without drop out regularization the accuracy is around 89.39% and with the drop out regularization the accuracy is around 90%.

Exercise 7: Specify the optimal values for the following parameters (the number of filters in the convolutional layer, the size of the convolutional kernel, the number of neurons in the dense hidden layer, the number of epochs used for training, the learning rate) that maximize the system accuracy.

I have tested with the several configuration, in most cases the system over-fit. The following are the best performance.

No of filters: 64

Kernel size: (3x3)

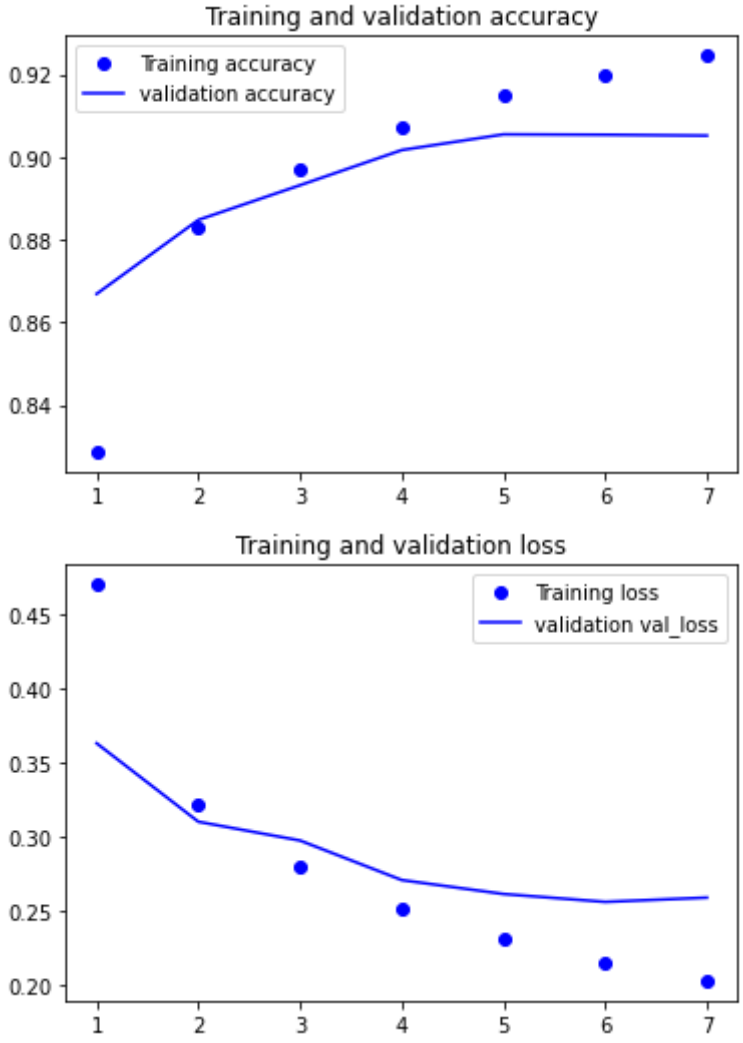
No neurons in the dense hidden layer: 250

epoch size: 7

learning rate: 0.001

Mohamed ABDUL GAFOOR

The following figure shows the training accuracy, validation accuracy and training loss and the validation loss.



Exercise 8: Once fit, we can save the final model (architecture and weights) to an *.h5 file by calling the *save()* function on the model and pass in the selected filename (`model.save('Fashion_MNIST_model.h5')`). We can use our saved model to make a prediction on new images.

The following screenshot shows the implementation. Here we reshape the tensor for the classification (reshaped tensor (1, 28, 28, 1)).

```
def load_fun(img):
  img = cv2.resize(img, (28,28))
  img_array = image.img_to_array(img)
  img_batch = np.expand_dims(img_array, axis=0)

  return img_batch

img = load_fun(img)
```

The following snippet shows the prediction. Here we can see the class is belongs to **2.**

```
model = load_model('Fashion_MNIST_model.h5')
```

```
pred = model.predict_classes(img)
print(pred)
```

```
[2]
```