Report Assignment #1

Sohaib Ali Gill

MSDS-19050

We were given a classification task to predict the numbers written on the images. MNIST numbers dataset was used, which has 60000 training images and 10,000 testing images. We were asked to implement a Neural Network to predict the numbers, first we train the model on the dataset then test the accuracy and I achieved 97% accuracy. I achieved this accuracy by manipulating different hyper parameters like batch size, learning rate, epochs etc. Report was prepared using different models of different setting.

A)-

With normalization, setting mean to zero and variance to 0.5, results are improved significantly. As it is approved from the graphs that the training and testing accuracies are higher then the not normalized dataset,91% and 88%, respectively. Model learns more quickly, and the losses were decreasing in more steeper manners then the dataset in non-normalized dataset.

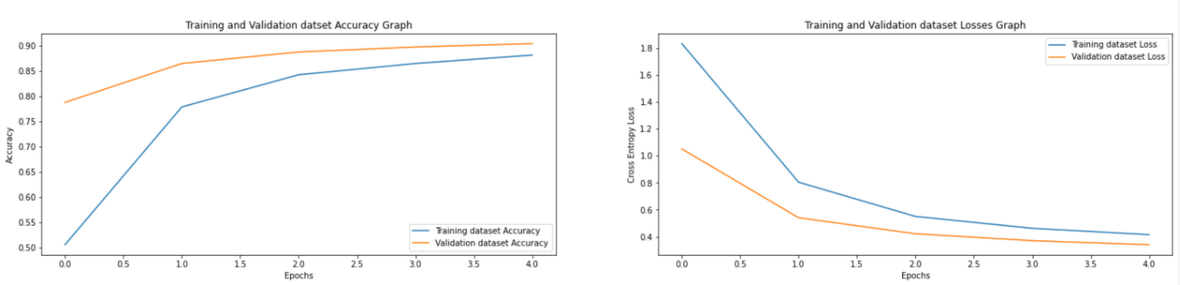
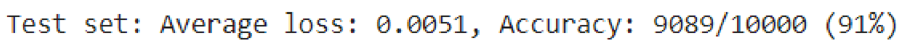


Figure 1 With Normalization



```
Test set: Average loss: 0.0051, Accuracy: 9089/10000 (91%)
```

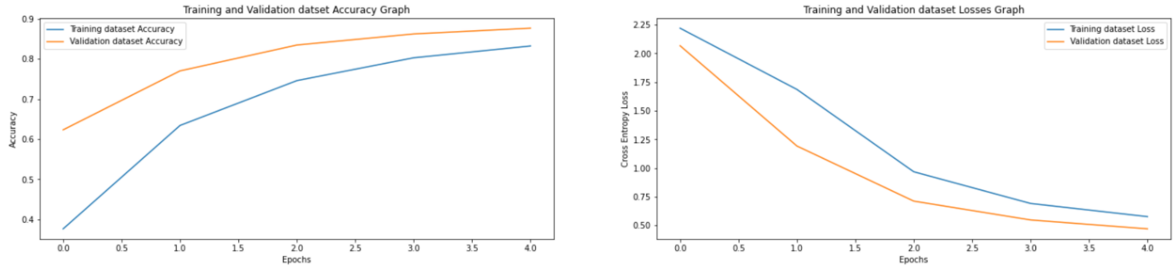
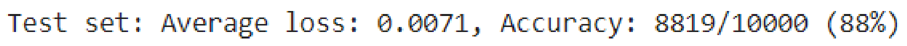


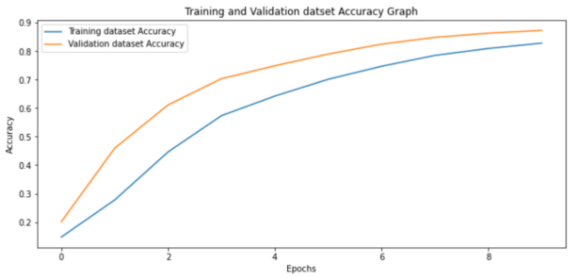
Figure 2 Without Normalization

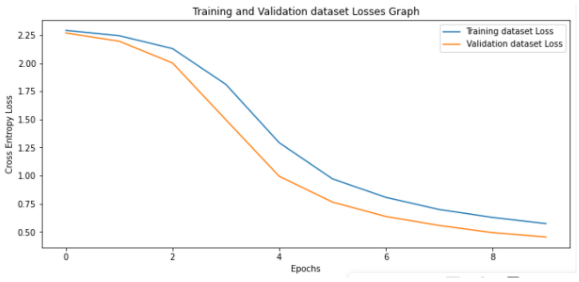


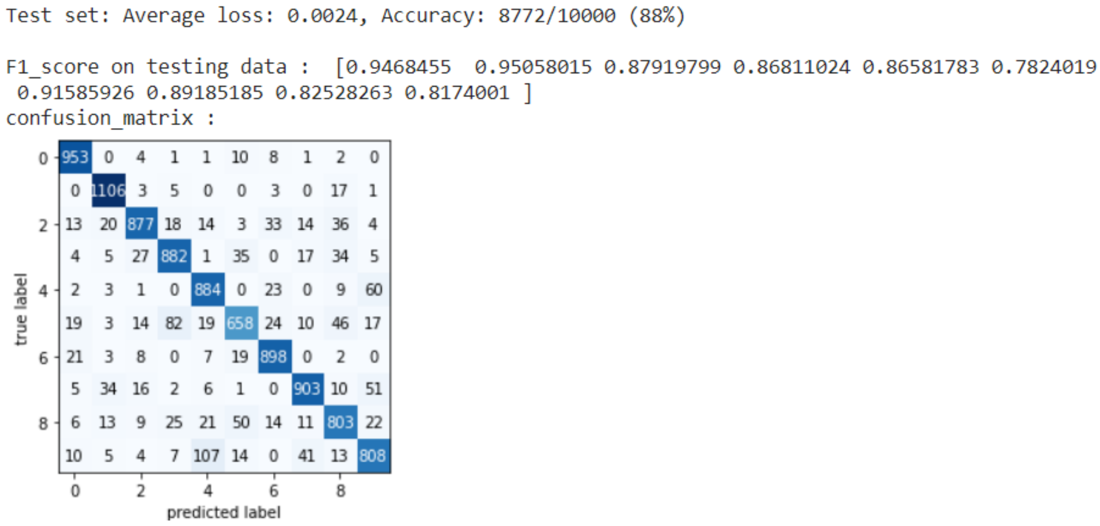
```
Test set: Average loss: 0.0071, Accuracy: 8819/10000 (88%)
```

B)-

By increasing the number of hidden layers and the number of neurons in the hidden layers and the batch size, results may improved but that large network also requires more time to learn means more epochs to pass through the dataset, with 5 epochs and 3 hidden layers it gives, 88% accuracy and as shown in the graph it learns very slowly and losses were also decreases at slower rate.







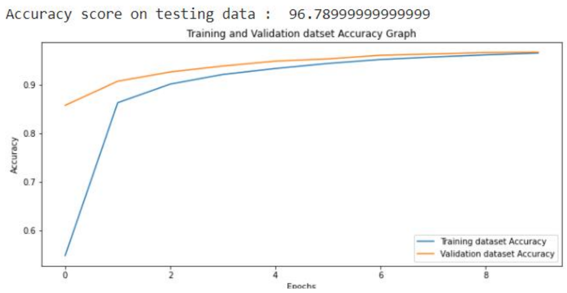
```
Test set: Average loss: 0.0024, Accuracy: 8772/10000 (88%)

F1_score on testing data :  [0.9468455  0.95058015 0.87919799 0.86811024 0.86581783 0.7824019
 0.91585926 0.89185185 0.82528263 0.8174001 ]
confusion_matrix :
```

true label

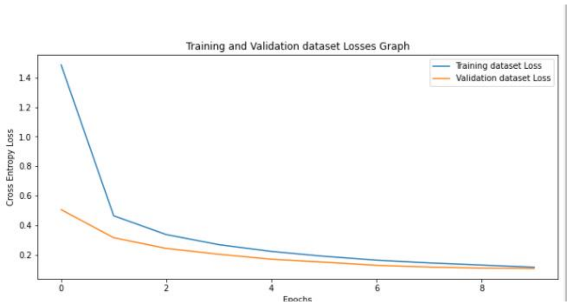
	0		2		4		6		8	
0	953	0	4	1	1	10	8	1	2	0
	0	1106	3	5	0	0	3	0	17	1
2	13	20	877	18	14	3	33	14	36	4
	4	5	27	882	1	35	0	17	34	5
4	2	3	1	0	884	0	23	0	9	60
	19	3	14	82	19	658	24	10	46	17
6	21	3	8	0	7	19	898	0	2	0
	5	34	16	2	6	1	0	903	10	51
8	6	13	9	25	21	50	14	11	803	22
	10	5	4	7	107	14	0	41	13	808

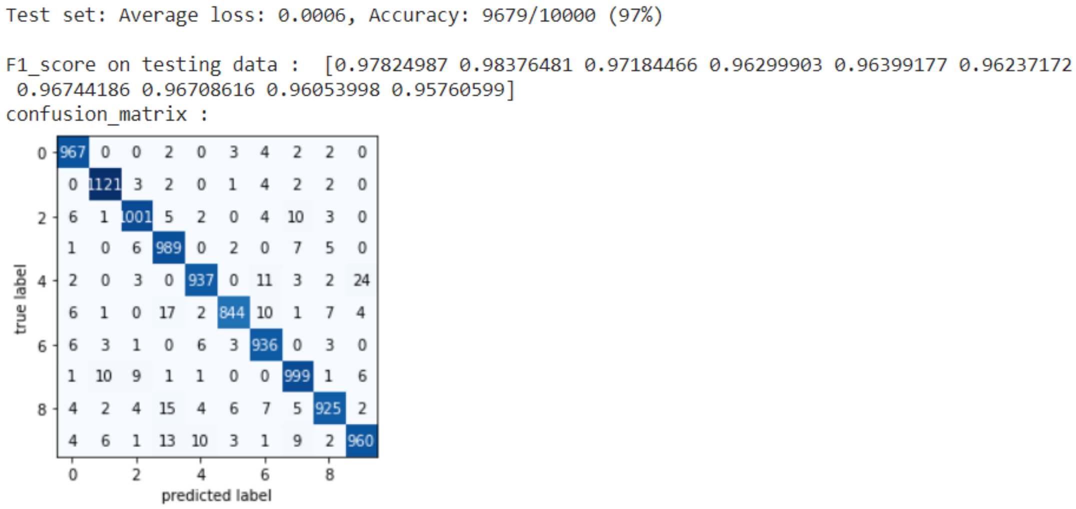
predicted label

As I increased the no of epochs to 10 and also increased the learning rate from 0.0005 to 0.005, then the accuracy jumped and lied at 97%. Model’s accuracy also jumped high and losses touched the rock bottom immediately, as I increased the learning rate.



```
Accuracy score on testing data :  96.78999999999999
```

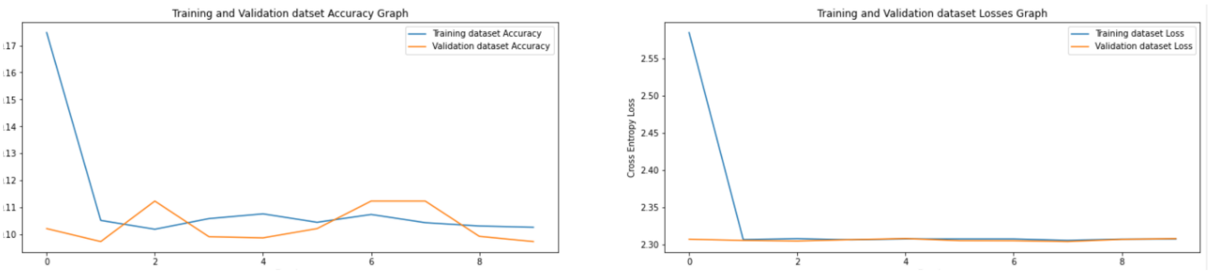




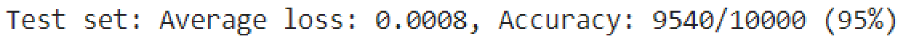
```
Test set: Average loss: 0.0006, Accuracy: 9679/10000 (97%)

F1_score on testing data :  [0.97824987 0.98376481 0.97184466 0.96299903 0.96399177 0.96237172
 0.96744186 0.96708616 0.96053998 0.95760599]
confusion_matrix :
```

	0		2		4		6		8	
0	967	0	0	2	0	3	4	2	2	0
	0	1121	3	2	0	1	4	2	2	0
2	6	1	1001	5	2	0	4	10	3	0
	1	0	6	989	0	2	0	7	5	0
4	2	0	3	0	937	0	11	3	2	24
	6	1	0	17	2	844	10	1	7	4
6	6	3	1	0	6	3	936	0	3	0
	1	10	9	1	1	0	0	999	1	6
8	4	2	4	15	4	6	7	5	925	2
	4	6	1	13	10	3	1	9	2	960

But its not all time case, if I increased more learning rate too much then the model does not learn anything, learning curve will just fluctuate as shown in below figure.



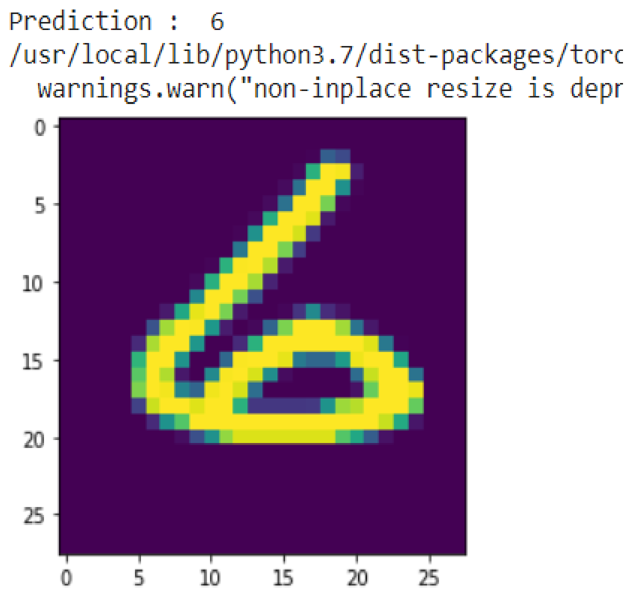
Test accuracy also decrease by decreasing the size of the training set to half because network got less datapoints to learn the features.



```
Test set: Average loss: 0.0008, Accuracy: 9540/10000 (95%)
```

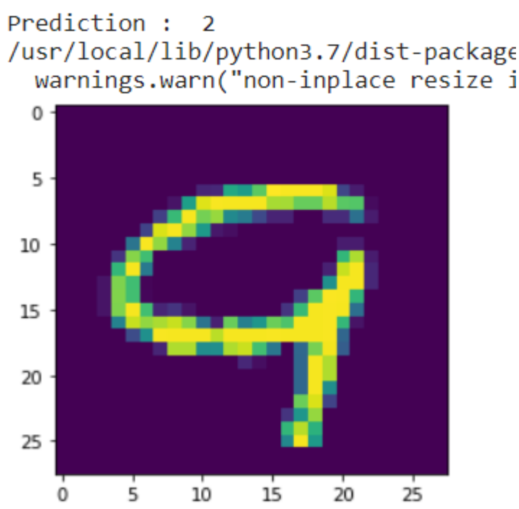
C)-

In most of the cases, model predict the correct number because model has high accuracy, because we have layers of large neuron so it can learn features more accurately. But in some case model predict wrong. As the example shown below, image looks like bit rotated ‘2’ , may be that’s why mode predicted 9 as 2 and in other example 4 looks like 9 , both has same basic features that’s why model predicted it wrong.

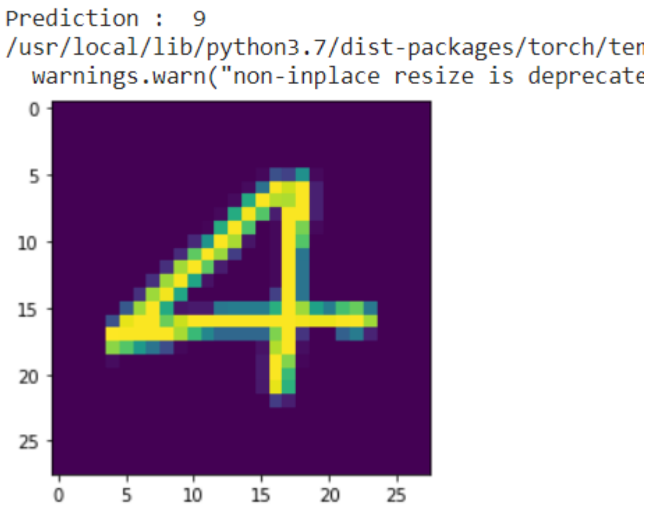


```
Prediction :  6
/usr/local/lib/python3.7/dist-packages/torc
  warnings.warn("non-inplace resize is depr
```

Figure 3 Correct prediction



```
Prediction :  2
/usr/local/lib/python3.7/dist-package
  warnings.warn("non-inplace resize i
```

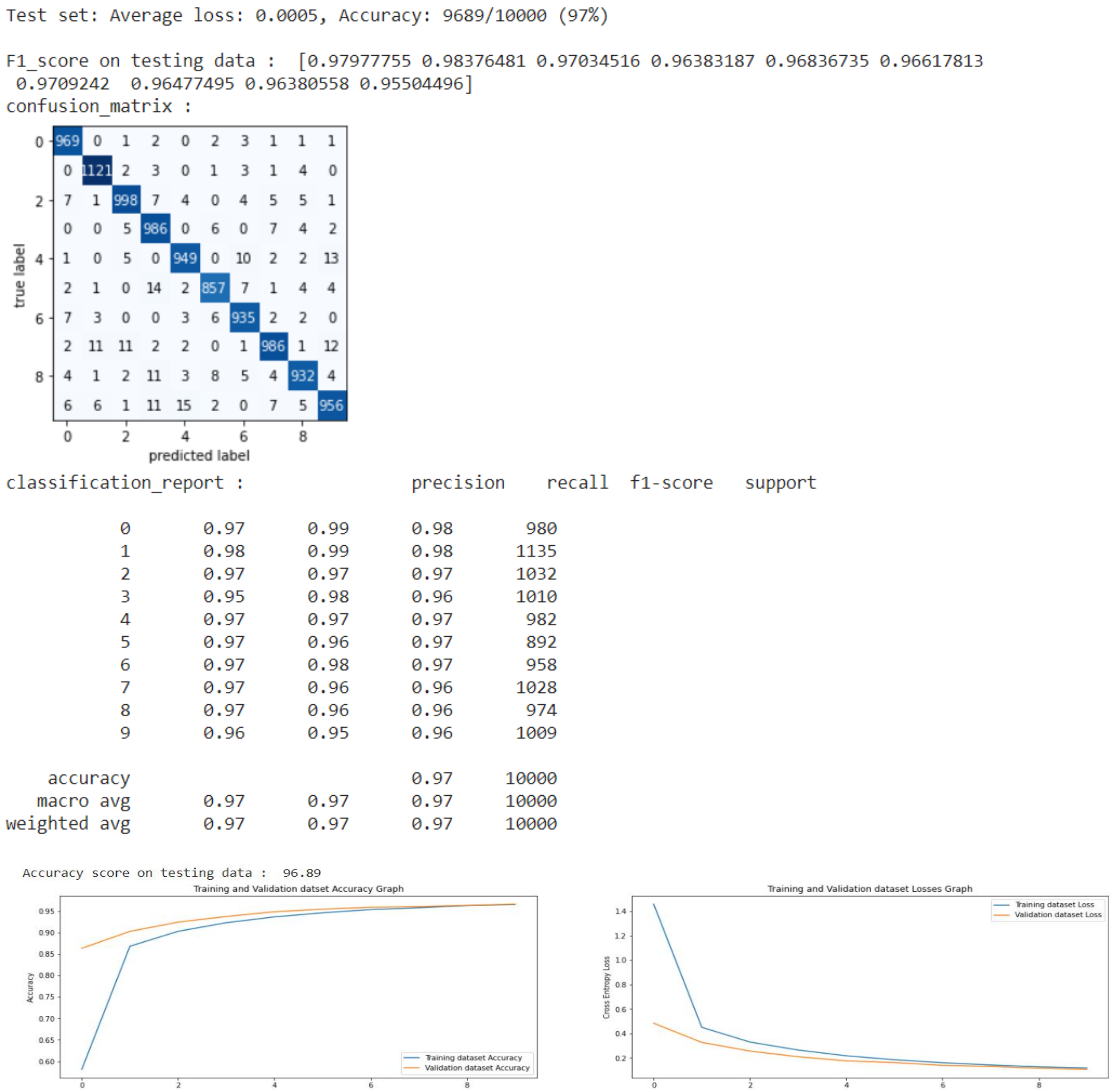


```
Prediction :  9
/usr/local/lib/python3.7/dist-packages/torch/ten
  warnings.warn("non-inplace resize is deprecate
```

Figure 4 wrong prediction

Figure 5 Wrong prediction

Confusion matrix, Recall, support and F1 score are displayed below with accuracy and loss curves.



```
Test set: Average loss: 0.0005, Accuracy: 9689/10000 (97%)

F1_score on testing data :  [0.97977755 0.98376481 0.97034516 0.96383187 0.96836735 0.96617813
 0.9709242  0.96477495 0.96380558 0.95504496]
confusion_matrix :
```

true label										
0	969	0	1	2	0	2	3	1	1	1
	0	1121	2	3	0	1	3	1	4	0
2	7	1	998	7	4	0	4	5	5	1
	0	0	5	986	0	6	0	7	4	2
4	1	0	5	0	949	0	10	2	2	13
	2	1	0	14	2	857	7	1	4	4
6	7	3	0	0	3	6	935	2	2	0
	2	11	11	2	2	0	1	986	1	12
8	4	1	2	11	3	8	5	4	932	4
	6	6	1	11	15	2	0	7	5	956
predicted label	0		2		4		6		8	

```
classification_report :               precision    recall  f1-score   support

           0       0.97      0.99      0.98       980
           1       0.98      0.99      0.98      1135
           2       0.97      0.97      0.97      1032
           3       0.95      0.98      0.96      1010
           4       0.97      0.97      0.97       982
           5       0.97      0.96      0.97       892
           6       0.97      0.98      0.97       958
           7       0.97      0.96      0.96      1028
           8       0.97      0.96      0.96       974
           9       0.96      0.95      0.96      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000

  Accuracy score on testing data :  96.89
```

Training and Validation datset Accuracy Graph
Accuracy
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
2
4
6
8
Training dataset Accuracy
Validation dataset Accuracy
Training and Validation dataset Losses Graph
Cross Entropy Loss
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
2
4
6
8
Training dataset Loss
Validation dataset Loss

D)-

From this assignment, I have learned how to create the neural network how it learn and how to set the different hyper parameters which are essential for the NN to learn better. I have learned that there is no specific procedure to select the number of layers and no of neurons in the network, its just a hit and trial process and where you find better results that's the most suitable network for your data. I have also learned the importance of learning rate in the training process, if we increase the learning rate the model will possibly learn nothing or less but model would be trained in less time but for accurate result you must select a moderate learning rate. I have also learned that bigger network is difficult to learn and for bigger network you should have bigger dataset otherwise the advance network layers does not learn anything and those layers are in vein, just slow the learning process. Suitable number of layers and neurons must be selected according to the complexity oof your dataset not too small, not too large.

E)-

When you have small network obviously it take less time to train the model, but it also affects the your prediction accuracy. When I used smaller network of 2 or 3 hidden layers it took 5-7 mins for 10 epochs and when I used 5 or 6 hidden layers it took approximately 12 mins to train the model. Same with case of CPU and GPU, deep learning models need more computational power to train the models because it has millions of parameters/features to learn so if the data is huge then it would take weeks on simple CPU to learn. But as we know GPUs are more fast, and they perform well on images that’s why GPUs are recommended for Deep Learning Model trainings.