[CM3]

Model 1 - Run time performance

In [ ]:
```
pred_start_time = time.time()
test_loss, test_acc = model.eval(loader = test_loader, BATCH_SIZE = 1)
pred_end_time = time.time()
print("Prediction time for Model 1 is ",(pred_end_time-pred_start_time)/l
print("Training time for Model 1 is ",training_time_tot," seconds")
```

```
Prediction time for Model 1 is  0.07086455821990967  seconds per sample
Training time for Model 1 is  102.02685689926147  seconds
```

Model 2 - Run time performance

In [ ]:
```
pred_start_time = time.time()
test_loss, test_acc = model2.eval(loader = test_loader, BATCH_SIZE = 1)
pred_end_time = time.time()
print("Prediction time for Model 2 is ",(pred_end_time-pred_start_time)/l
print("Training time for Model 2 is ",training_time_tot," seconds")
```

```
Prediction time for Model 2 is  0.04983115196228027  seconds per sample
Training time for Model 2 is  130.02915930747986  seconds
```

Execution Time: Since the models are relatively simpler, the run time performance was decent. Model 1 trained in 102 seconds while model 2 trained in 130 seconds. We can note that the training time increases with model complexity.

Runtime Performance depends on following factors:
Size of the model and design

- In terms of no of parameters, no of layers (depth). Convolution,despite requiring few parameters, is computationally intensive. So the number of convolution layers will have an impact on the runtime performance of the algorithm (both training andevaluation).

Batch size

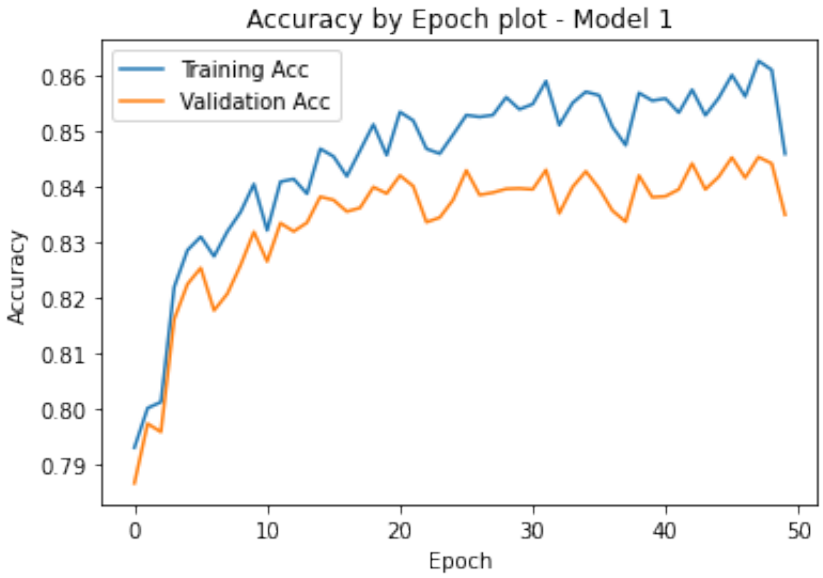
- This factor impacts training time and model convergence. We need to strike a balance in this hyper parameter as having high batch size will cause GPU memory overflow.

Hardware: Usage of GPU significantly boosts the run time permormance mainly due to ability to run things in parallel. In our execution, we are using google colab with GPU run time for hardware acceleration.

Model 1	Model 1	Model 2	Model 2	
Training per epoch	Testing per epoch	Training per epoch	Testing per epoch	
102s	0.07s	130s	0.04s	------

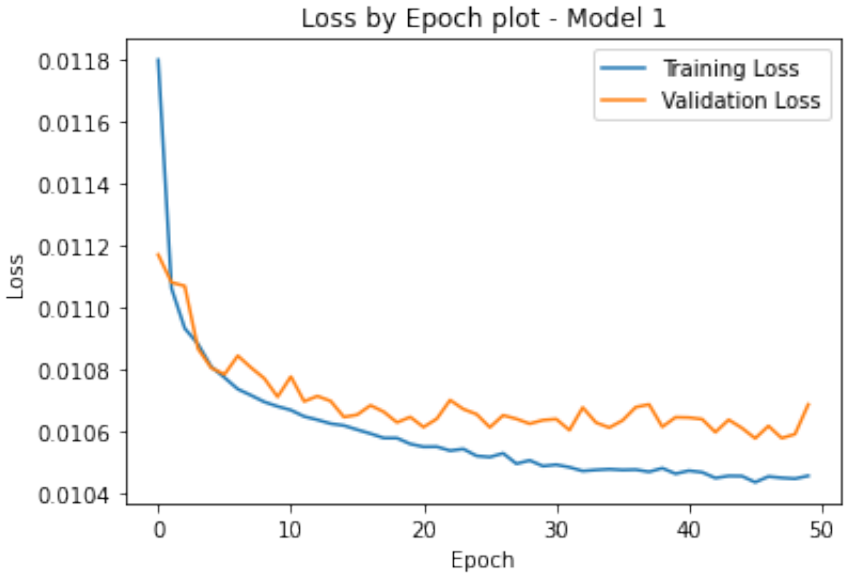
Epoch	Training Accuracy	Validation accuracy	Training Accuracy	Validation accuracy
10	0.79	0.82	0.79	0.845
20	0.84	0.835	0.85	0.855
30	0.85	0.838	0.86	0.857
40	0.85	0.84	0.88	0.86
50	0.86	0.83	0.89	0.85

In [ ]:
```
plt.plot(tr_acc, label = 'Training Acc')
plt.plot(np.array(val_acc)/100.0, label = 'Validation Acc')
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.title("Accuracy by Epoch plot - Model 1")
plt.legend()
plt.show()
```



In [ ]:
```
plt.plot(np.array(tr_loss)/100, label = 'Training Loss')
plt.plot(np.array(val_loss), label = 'Validation Loss')
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Loss by Epoch plot - Model 1")
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x7ff48b218d90>
```

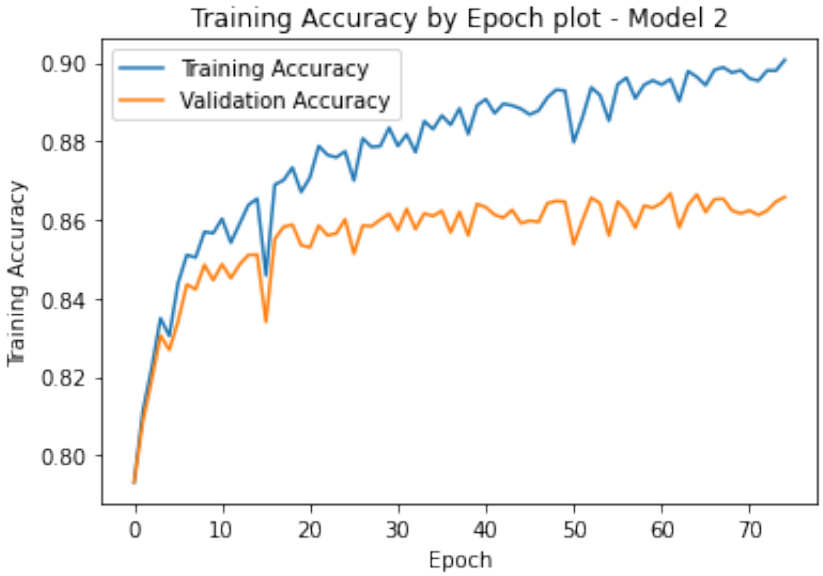


In [ ]:
```
test_loss,test_acc = model.eval(loader=test_loader,BATCH_SIZE=100)
print("Test loss for Model 1 is ", test_loss)
print("Test accuracy for Model 1 is ", test_acc)
```

```
Test loss for Model 1 is  0.010652118769183854
Test accuracy for Model 1 is  83.85
```

In [ ]:
```
plt.plot(tr_acc2, label = 'Training Accuracy')
plt.plot(val_acc2, label = 'Validation Accuracy')
plt.xlabel("Epoch")
plt.ylabel("Training Accuracy")
plt.title("Training Accuracy by Epoch plot - Model 2")
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x7ff48b4c86d0>
```



In [ ]:
```
plt.plot(np.array(tr_loss2)/100, label = 'Training Loss')
plt.plot(val_loss2, label = 'Validation Loss')
plt.xlabel("Epoch")
plt.ylabel("Training Loss")
plt.title("Training Loss by Epoch plot - Model 2")
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x7ff48b3c3350>
```

In [ ]:
```
test_loss,test_acc = model2.eval(loader=test_loader,BATCH_SIZE=100)
print("Test loss for Model 2 is ", test_loss)
print("Test accuracy for Model 2 is ", test_acc)
```

```
Test loss for Model 2 is  0.010438133199435765
Test accuracy for Model 2 is  0.86
```