Project Development Phase
Model Performance Test

Date	10 November 2022
Team ID	PNT2022TMID35610
Project Name	Project - A Novel Method For Handwritten Digit Recognition System.
Maximum Marks	10 Marks

Model Performance Testing:

S.No.	Parameter	Values	Screenshot
1.	Model Summary		[12]: `model.summary()` Model: "sequential_2" Layer (type) — Output Shape — Param # conv2d_4 (Conv2D) — (None, 26, 26, 28) — 280 max_pooling2d_4 (MaxPooling2 — (None, 13, 13, 28) — 0 conv2d_5 (Conv2D) — (None, 11, 11, 28) — 7084 max_pooling2d_5 (MaxPooling2 — (None, 5, 5, 28) — 0 flatten_2 (Flatten) — (None, 700) — 0 dense_6 (Dense) — (None, 512) — 358912 dropout_4 (Dropout) — (None, 512) — 0 dense_7 (Dense) — (None, 128) — 65664 dropout_5 (Dropout) — (None, 128) — 0 dense_8 (Dense) — (None, 10) — 1290 Total params: 433,230 Trainable params: 433,230 Non-trainable params: 0 + Code + Markdown

2.	Accuracy	Training Accuracy – 99.81% Validation Accuracy – 98.91% Testing Accuracy-99.04%	

```
def accuracy(x_train, y_train, model):

    loss,acc = model.evaluate(train_samples, train_labels,verbose=0)
    return acc

acc = accuracy(train_samples, train_labels, model)
print('Train accuracy is, ', acc*100, '%')
```

```
Train accuracy is,  99.81666803359985 %
```

	Validation Accuracy	Training Accuracy	Validation Loss	Training Loss	Epoch
0	0.978500	0.906852	0.074422	0.308299	1
1	0.987500	0.975963	0.047910	0.078035	2
2	0.987833	0.984593	0.043793	0.051941	3
3	0.988500	0.988389	0.041744	0.038774	4
4	0.989000	0.990778	0.041208	0.029460	5
5	0.990167	0.992185	0.036659	0.024544	6
6	0.988833	0.993389	0.037043	0.020551	7
7	0.989667	0.994778	0.039521	0.016402	8
8	0.991167	0.995241	0.041283	0.014158	9
9	0.990000	0.995148	0.037630	0.013622	10
10	0.991000	0.996463	0.038866	0.011439	11
11	0.991333	0.996315	0.036490	0.011447	12
12	0.992500	0.996574	0.037629	0.009888	13
13	0.991500	0.997574	0.043673	0.007542	14
14	0.991500	0.997167	0.042455	0.008863	15
15	0.990833	0.996944	0.041079	0.008768	16
16	0.990333	0.997389	0.053278	0.007346	17
17	0.990667	0.997370	0.046048	0.007472	18
18	0.992167	0.997426	0.041623	0.007897	19
19	0.989167	0.997370	0.050672	0.007211	20

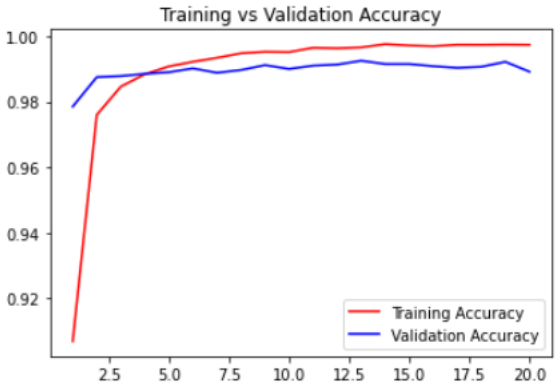
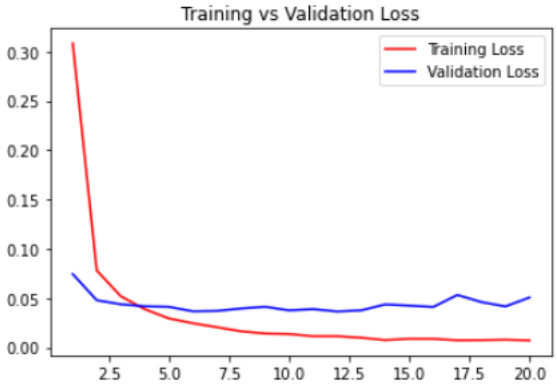
[48]:

```
def accuracy(x_test, y_test, model):

    loss,acc = model.evaluate(test_samples, test_labels,verbose=0)
    return acc

acc = accuracy(test_samples, test_labels, model)
print('Test accuracy is, ', acc*100, '%')
```

```
Test accuracy is,  99.04000163078308 %
```

			 Training vs Validation Accuracy 1.00 0.98 0.96 0.94 0.92 Training Accuracy Validation Accuracy 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0  Training vs Validation Loss 0.30 0.25 0.20 0.15 0.10 0.05 0.00 Training Loss Validation Loss 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0

3.	Confusion Matrix		[47]:
			`def create_confusion_matrix(true_labels, predicted_labels):`
			
			`  from sklearn.metrics import confusion_matrix`
			`  cm = confusion_matrix(true_labels.argmax(axis=1), predicted_labels.argmax(axis=1))`
			`  return cm`
			
			`cm = create_confusion_matrix((train_labels), (predict(train_samples)))`
			`print(cm)`
			
			`[[5917  0  2  1  0  0  0  0  1  2]`
			` [  0 6738  0  1  0  1  2  0  0  0]`
			` [  0  1 5956  0  0  0  0  0  1  0]`
			` [  0  0  1 6128  0  1  0  0  1  0]`
			` [  0  3  0  0 5815  0  1  3  1 19]`
			` [  0  0  1  7  0 5406  5  0  2  0]`
			` [  0  0  0  0  1  0 5916  0  1  0]`
			` [  0  4 16  6  0  0  0 6236  0  3]`
			` [  0  0  3  8  0  0  0  0 5840  0]`
			` [  1  0  0  4  1  3  0  0  2 5938]]`
			
			+ Code + Markdown
			
			The confusion matrix gives the performance of our model on a set of test data.

4.	Classification Report		[56]:
			`from sklearn.metrics import classification_report, confusion_matrix`
			
			`print(`
			`  f"Classification report for classifier :\n"`
			`  f"{classification_report(train_labels.argmax(axis=1), predict(train_samples).argmax(axis=1))}\n"`
			`)`
			
			`Classification report for classifier :`
			`              precision    recall  f1-score   support`
			
			`           0       1.00      1.00      1.00      5923`
			`           1       1.00      1.00      1.00      6742`
			`           2       1.00      1.00      1.00      5958`
			`           3       1.00      1.00      1.00      6131`
			`           4       1.00      1.00      1.00      5842`
			`           5       1.00      1.00      1.00      5421`
			`           6       1.00      1.00      1.00      5918`
			`           7       1.00      1.00      1.00      6265`
			`           8       1.00      1.00      1.00      5851`
			`           9       1.00      1.00      1.00      5949`
			
			`    accuracy                           1.00     60000`
			`   macro avg       1.00      1.00      1.00     60000`
			`weighted avg       1.00      1.00      1.00     60000`