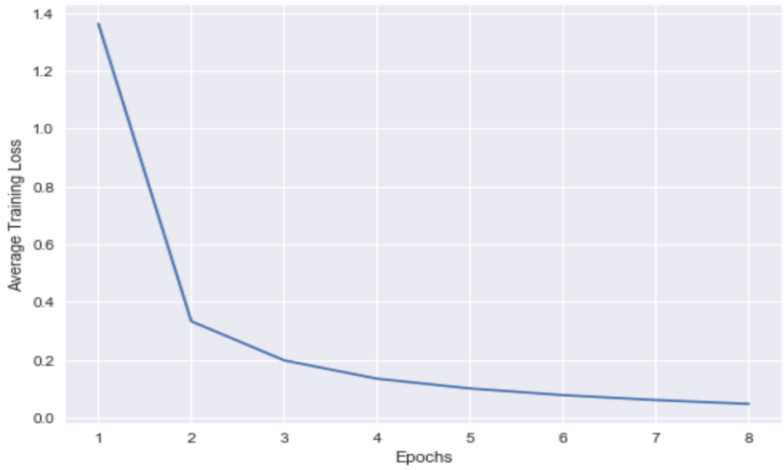
COL341
Assignment A2.2 Report

Ishaan Watts
2019PH10629

Part A

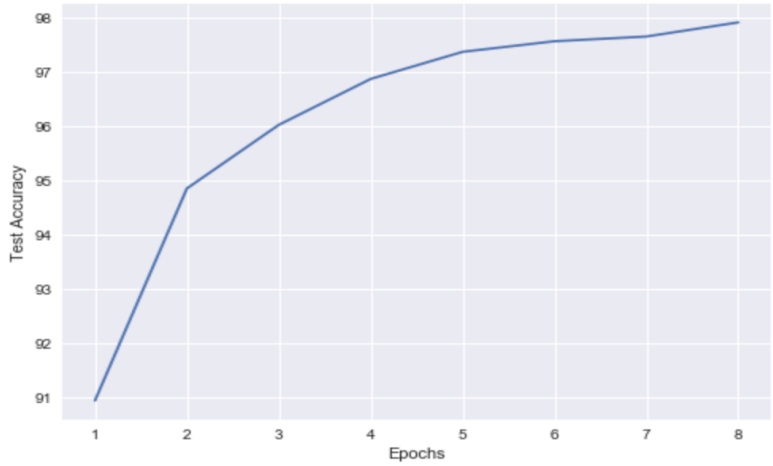
1. Average Training Loss




```
loss
```

```
array([1.36317676, 0.33332235, 0.19803374, 0.13449777, 0.10074657,
       0.07772532, 0.06065874, 0.04740315])
```

2. Testing Accuracy




```
acc
```

```
array([90.93478261, 94.84782609, 96.02173913, 96.86956522, 97.36956522,
       97.56521739, 97.65217391, 97.91304348])
```

3. Results

After 8 Epochs:

Public Test Accuracy --> **97.913%**
Average Training Loss --> **0.04740315**

4. Comparison with A2.1

a) In assignment A2.1 a neural network was used on Devanagari Dataset. After all the hyperparameter tuning and selecting the best architecture the results were as follows:

Test Accuracy = 95.6957, Epochs = 29
Program runs under 5min on CPU

b) In assignment A2.2a, an inefficient implementation of a CNN gave significantly better results as follows:

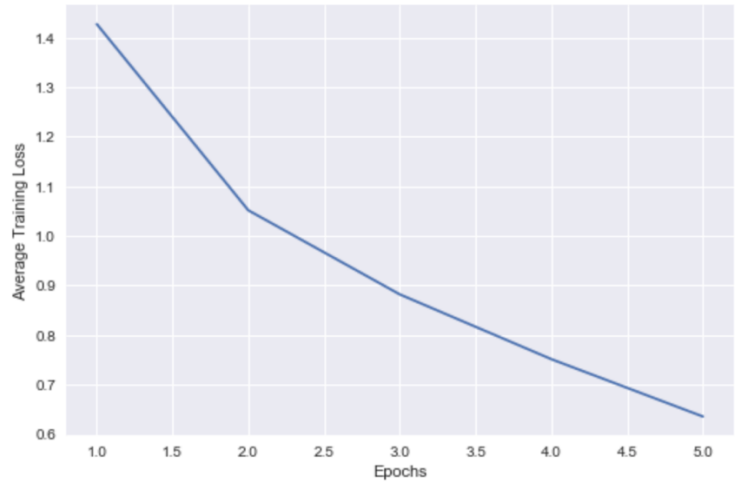
Test Accuracy = 97.913%, Epochs = 8
Programs runs under 2min on GPU

5. Conclusion

Clearly the untuned CNN architecture gave much better results than the naïve neural network in less training time.

Part B

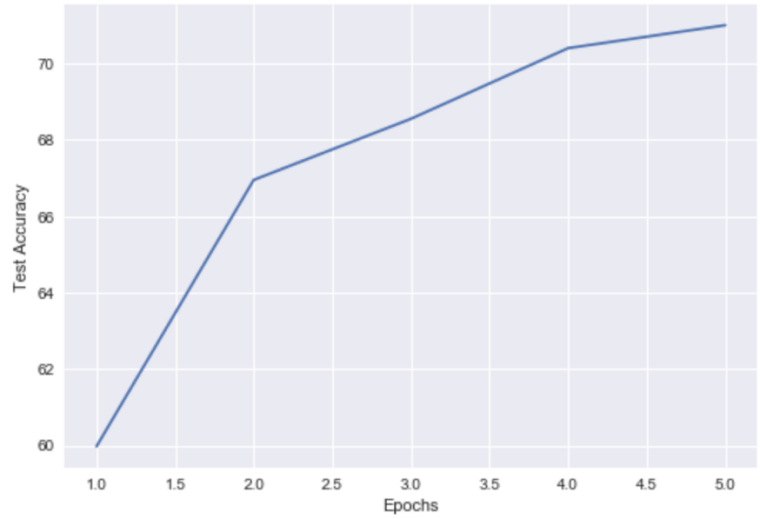
1. Average Training Loss




```
loss
```

```
array([1.42684858, 1.05149074, 0.88194339, 0.75140861, 0.63578974])
```

2. Testing Accuracy




```
acc
```

```
array([59.975, 66.95 , 68.55 , 70.4  , 71.   ])
```

3. Results

After 5 Epochs:

Public Test Accuracy --> **71%**
Average Training Loss --> **0.63578974**

Part C

The CNN architecture used in part b was quite inefficient. I read some papers and used the concepts of AlexNet (use of smaller filters
and relu) and VGG16 (stacked convolutional layers) to construct my architecture.
A lot of experimentation was done with each variant was run on Google Colab GPU for 20 epochs.

Main Architecture 1

Modification of architecture in part B with 3 double convolutional blocks with padding.

S.No.	Layer	In Channels	Out Channels	Kernel Size	Stride	Padding
1.	Conv2D	3	32	3	1	1
2.	BatchNorm2D	32		-	-	-
3.	Relu	-	-	-	-	-
4.	Conv2D	32	64	3	1	1
5.	Relu	-	-	-	-	-
6.	MaxPool2D	-	-	2	2	-
7.	Conv2D	64	128	3	1	1
8.	BatchNorm2D	128		-	-	-
9.	Relu	-	-	-	-	-
10.	Conv2D	128	128	3	1	1
11.	Relu	-	-	-	-	-
12.	MaxPool2D	-	-	2	2	-
13.	Conv2D	128	256	3	1	1
14.	BatchNorm2D	256		-	-	-
15.	Relu	-	-	-	-	-
16.	Conv2D	256	256	3	1	1
17.	Relu	-	-	-	-	-
18.	MaxPool2D	-	-	2	2	-
19.	Dropout	p = 0.1				
20.	Linear	4096	420	-	-	-
21.	Relu	-	-	-	-	-
22.	Linear	420	256	-	-	-

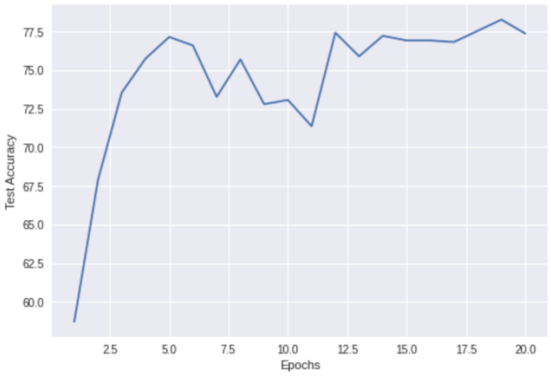
23.	Relu	-	-	-	-	-
24.	Dropout	p = 0.1				
25.	Linear	256	10	-	-	-

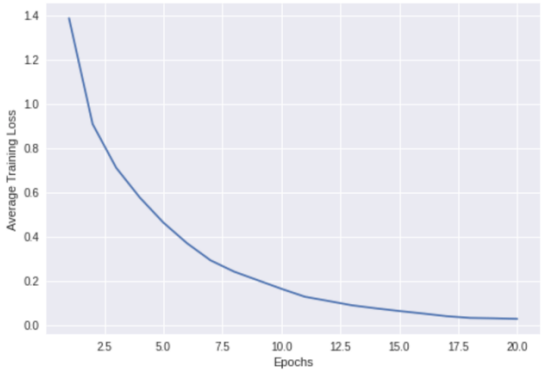
Model Parameters = 2,957,998

Exp 1 – New Architecture

Note: I had also tried using batch norm after all the convolutional layers which resulted in reduced accuracy. Removing one batch norm before maxpool increased accuracy significantly.

Optimizer = Adam, Lr = 1e-4, Batch Size = 200, Activation Relu





Training Loss = 0.0272, Max Testing Accuracy = 78.275.
As we can see the accuracy increased significantly from 71% in part B

Exp 2 – Data Augmentation

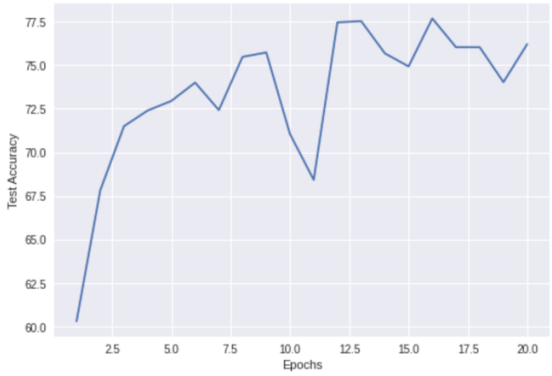
Use of Data Augmentation techniques

Two sets of normalization transforms were used

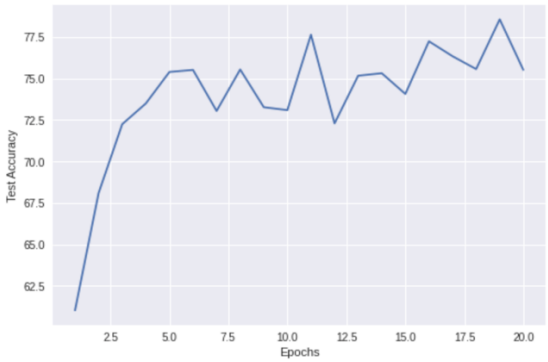
1. (0.5, 0.5, 0.5), (0.5, 0.5, 0.5)
Best Test Accuracy --> 77.675
Training Loss --> 0.0316

2. (0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)
Best Test Accuracy --> 78.575
Training Loss --> 0.0261

Norm-1



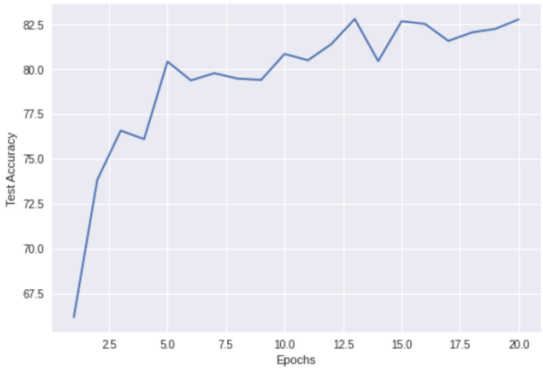
Norm-2

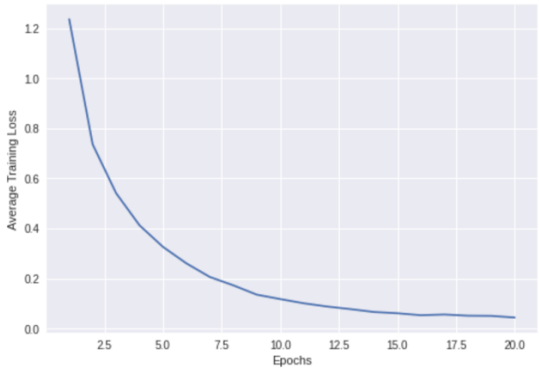


Hence, I chose Norm 2 for Data Augmentation.

Exp 3 - Learning Rate

Now I wanted to experiment with learning rate.
I tried Adam with Learning rate = 1e-3.



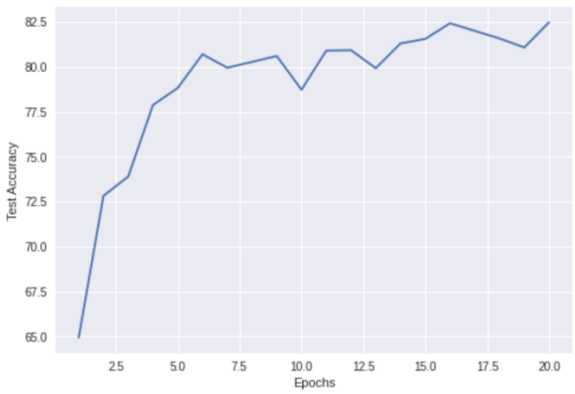


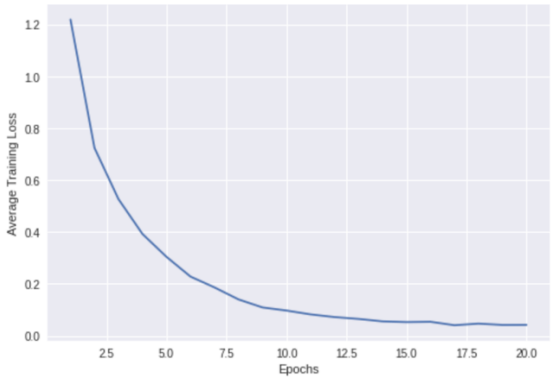
Training Loss = 0.0441, Max Testing Accuracy = 82.8

Tuning the learning rate greatly increased the accuracy and hence I chose learning rate as 1e-3 for Adam.

Exp 4 – Activation

I also experimented with activation by trying out leaky relu with rest parameters fixed as in Exp 3.
Max Test Accuracy --> 82.475
Training Loss --> 0.0418
These results are quite similar to those by relu.
Hence, I am using relu only which performed marginally better.

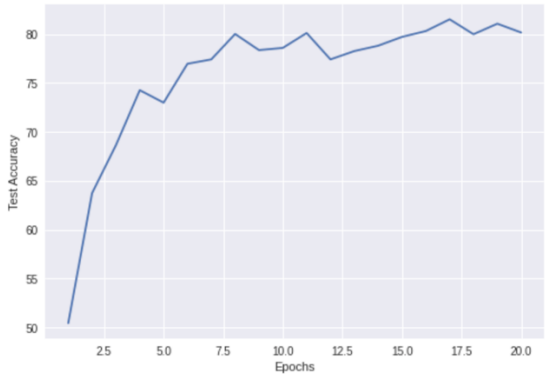




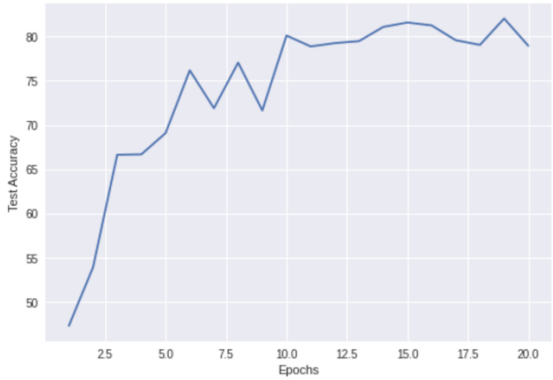
Exp 5 – Optimizer

I experimented with RMS prop with both relu and leaky relu with learning rate fixed at 1e-3 and weight decay 1e-6.

Leaky Relu



Relu



1. Leaky Relu
Best Test Accuracy --> 81.525
Training Loss --> 0.0525

2. Relu
Best Test Accuracy --> 82.025
Training Loss --> 0.0121

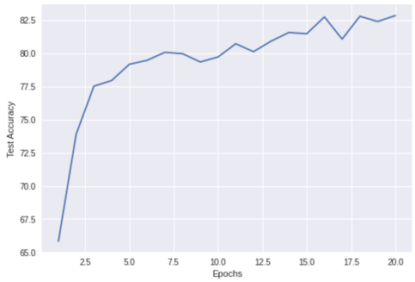
We can easily notice the decrease in test accuracy and increase in training loss. Therefore, I chose Adam with lr = 1e-3 and relu.

Exp 6 – Batch Size and Weight Decay

Since I have decided on Adam, I now experimented with the weight decay and batch size.

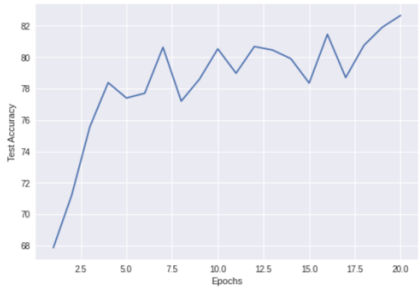
1. Batch size = 64,
   Weight decay = 0

Best Test Accuracy --> 82.85
Training Loss --> 0.0751



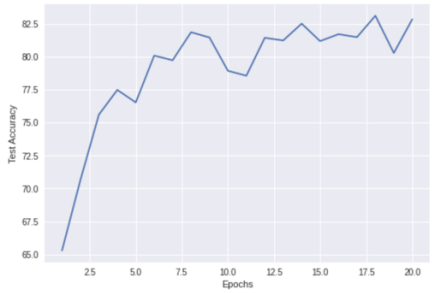
2. Batch size = 64
   Weight decay = 3e-4

Best Test Accuracy --> 82.65
Training Loss --> 0.151



3. Batch size = 256
   Weight decay = 3e-4

Best Test Accuracy --> 83.1
Training Loss --> 0.0922



From this I concluded weight decay was beneficial with larger batch size and hence I am using batch size = 200 and weight decay 3e-4 with Adam.

All these experiments resulted into little increase in test accuracy. Now I decided to modify the architecture keeping the above best parameters fixed.

Exp 7 – Dropout

I tried inserting dropout layers in the model in different ways.

1. Dropout after Convolutional Block2 with p = 0.05
2. Doubling the value of all dropout layers, i.e., p = 0.1 for Convolutional Block2 and p = 0.2 for fully connected layers.
3. Adding dropout layer with p = 0.1 in all Convolutional Blocks and p = 0.2 for Fully Connected layers.
4. Adding dropout layer with p = 0.05 in all Convolutional Blocks and p = 0.2 for Fully Connected layers.

Architecture for variant 4 (best out of these)

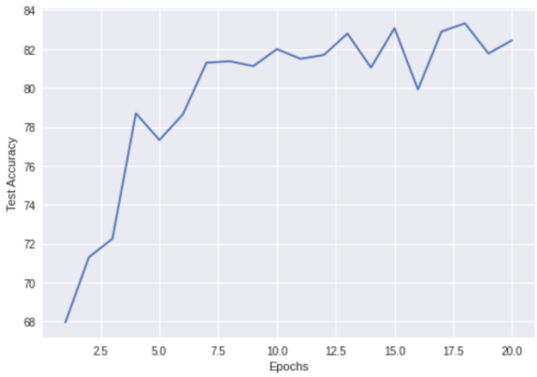
S.No.	Layer	In Channels	Out Channels	Kernel Size	Stride	Padding
1.	Conv2D	3	32	3	1	1
2.	BatchNorm2D	32		-	-	-
3.	Relu	-	-	-	-	-
4.	Conv2D	32	64	3	1	1
5.	Relu	-	-	-	-	-
6.	MaxPool2D	-	-	2	2	-
7.	Dropout2D	p = 0.05				
8.	Conv2D	64	128	3	1	1
9.	BatchNorm2D	128		-	-	-
10.	Relu	-	-	-	-	-
11.	Conv2D	128	128	3	1	1
12.	Relu	-	-	-	-	-
13.	MaxPool2D	-	-	2	2	-
14.	Dropout2D	p = 0.05				
15.	Conv2D	128	256	3	1	1
16.	BatchNorm2D	256		-	-	-
17.	Relu	-	-	-	-	-
18.	Conv2D	256	256	3	1	1
19.	Relu	-	-	-	-	-
20.	MaxPool2D	-	-	2	2	-
21.	Dropout2D	p = 0.05				
22.	Dropout	p = 0.2				
23.	Linear	4096	420	-	-	-
24.	Relu	-	-	-	-	-
25.	Linear	420	256	-	-	-
26.	Relu	-	-	-	-	-
27.	Dropout	p = 0.2				
28.	Linear	256	10	-	-	-

Model Parameters = 2,957,998

All the 4 variants resulted in a significant increase in test accuracy.

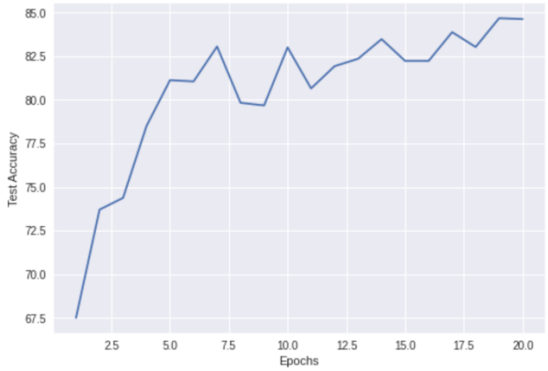
1. Dropout after Convolutional Block2 with p = 0.05

Best Test Accuracy = 83.325
Train Loss = 0.1078



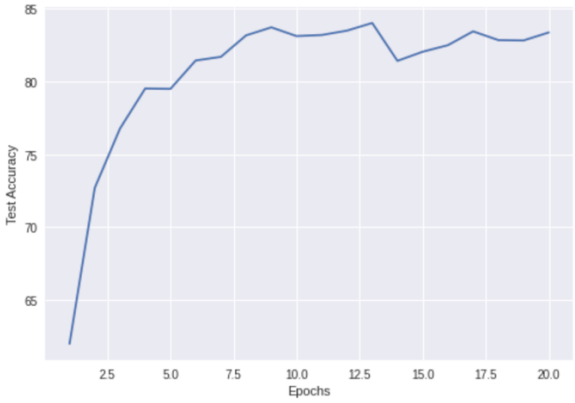
2. Doubling the value of all dropout layers, i.e., p = 0.1 for Convolutional Block2 and p = 0.2 for fully connected layers.

Best Test Accuracy = 84.675
Train Loss = 0.1333



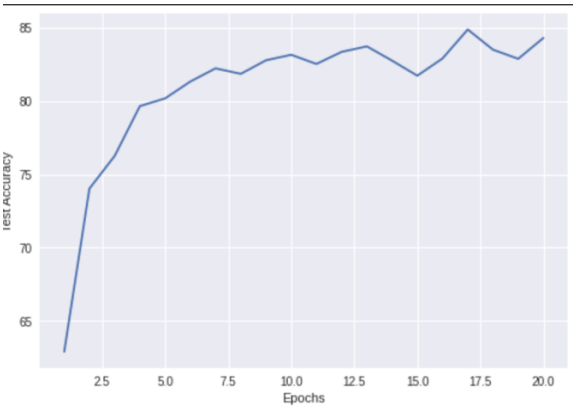
3. Adding dropout layer with p = 0.1 in all Convolutional Blocks and p = 0.2 for Fully Connected layers.

Best Test Accuracy = 84.025
Train Loss = 0.1814



4. Adding dropout layer with p = 0.05 in all Convolutional Blocks and p = 0.2 for Fully Connected layers.

Best Test Accuracy = 84.875
Train Loss = 0.1420



Till now the 4th variant was the best and was further experimented upon.

Exp 8 – Architecture

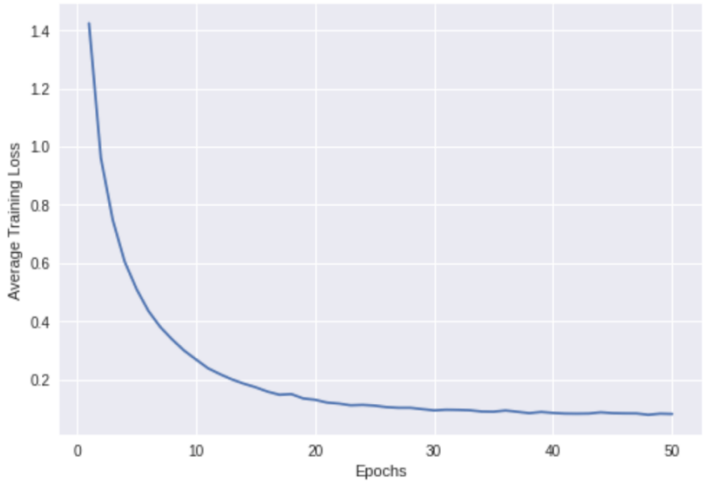
Using all the results from before I constructed a new architecture with an additional convolutional block.
Program ran for 50 epochs.

Best Architecture

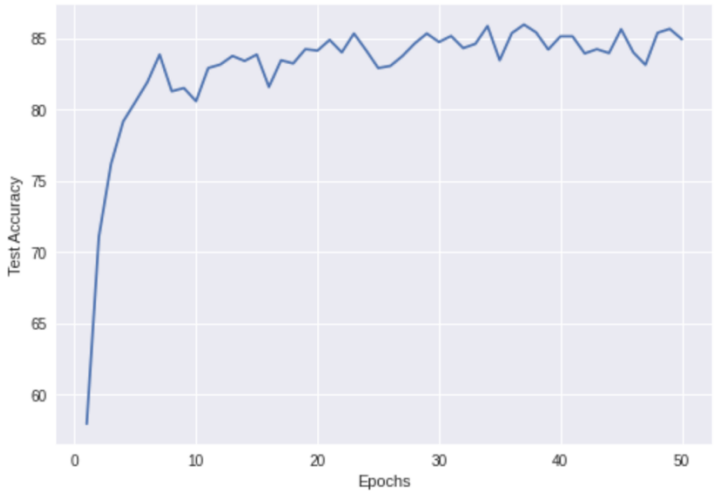
S.No.	Layer	In Channels	Out Channels	Kernel Size	Stride	Padding
1.	Conv2D	3	32	3	1	1
2.	BatchNorm2D	32		-	-	-
3.	Relu	-	-	-	-	-
4.	Conv2D	32	64	3	1	1
5.	Relu	-	-	-	-	-
6.	MaxPool2D	-	-	2	2	-
7.	Dropout2D	p = 0.05				
8.	Conv2D	64	128	3	1	1
9.	BatchNorm2D	128		-	-	-
10.	Relu	-	-	-	-	-
11.	Conv2D	128	128	3	1	1
12.	Relu	-	-	-	-	-
13.	MaxPool2D	-	-	2	2	-
14.	Dropout2D	p = 0.05				
15.	Conv2D	128	256	3	1	1
16.	BatchNorm2D	256		-	-	-
17.	Relu	-	-	-	-	-
18.	Conv2D	256	256	3	1	1
19.	Relu	-	-	-	-	-
20.	MaxPool2D	-	-	2	2	-
21.	Dropout2D	p = 0.05				
22.	Conv2D	256	256	3	1	1
23.	BatchNorm2D	256		-	-	-
24.	Relu	-	-	-	-	-
25.	Conv2D	256	256	3	1	1
26.	Relu	-	-	-	-	-
27.	MaxPool2D	-	-	2	2	-
28.	Dropout2D	p = 0.05				

29.	Dropout	p = 0.2				
30.	Linear	1024	512	-	-	-
31.	Relu	-	-	-	-	-
32.	Linear	512	256	-	-	-
33.	Relu	-	-	-	-	-
34.	Dropout	p = 0.2				
35.	Linear	256	10	-	-	-

Training Loss



Test Accuracy



Results:

Training loss = 0.080626
Best Test Accuracy = 85.95

Final Parameters:

Optimizer --> Adam
Learning Rate --> 1e-3, Weight Decay --> 3e-4
Batch Size --> 200
Activation --> Relu
Model Parameters --> 2,966,282

Test Accuracy increases as we increase the epochs.

Supplementary Values for Exp 8:

Training loss =
[1.4236969814697902, 0.9609348827600479, 0.7471070220073064, 0.6039874332149824, 0.5097151017189026, 0.4342973511417707, 0.37918516630927723, 0.3363386241098245, 0.2982403642932574, 0.2677162844936053, 0.23783687993884087, 0.21757946118712426, 0.19956659585237502, 0.18479772369066874, 0.17242093443870543, 0.15747043805817762, 0.14671369589865207, 0.14898375660181046, 0.13399517878269157, 0.12941138328363497, 0.11976174432784319, 0.11625555627048016, 0.11037522396693628, 0.11183911240970094, 0.10905645267417033, 0.10395380757128199, 0.10174865478649736, 0.10173605888461074, 0.09721706949795286, 0.09282214765747389, 0.09512187870840232, 0.09453107391173642, 0.09330327662949761, 0.0888634519589444, 0.0882841898004214, 0.09255624625831843, 0.08803184765080611, 0.0833057590884467, 0.08751637270053228, 0.08405397316440939, 0.08207120878621936, 0.08169744967793426, 0.08216862642516692, 0.08616643204043309, 0.08335873417245845, 0.08268637841567397, 0.08248362230757872, 0.07771027451070646, 0.08172898976132274, 0.08062635052328308]
Test Accuracy =
[57.95, 71.075, 76.175, 79.15, 80.525, 81.925, 83.85, 81.275, 81.5, 80.575, 82.9, 83.15, 83.75, 83.4, 83.85, 81.575, 83.45, 83.225, 84.225, 84.125, 84.875, 84.0, 85.325, 84.15, 82.9, 83.05, 83.75, 84.625, 85.325, 84.725, 85.15, 84.3, 84.6, 85.85, 83.45, 85.35, 85.95, 85.4, 84.2, 85.125, 85.125, 83.925, 84.225, 83.95, 85.625, 84.0, 83.125, 85.375, 85.65, 84.925]