

# CMPT 412 Project 2 Report

**Ritika Goyal (301401516)**

I am using one lay day for assignment2

## Part 1: Improving BaseNet on CIFAR100

(Kaggle submission under name RitikaGoyal)

### Final Network

I used the reference from AlexNet model and the twerked different parameters to get accuracy on validation images around 63% with 64.3% score on Kaggle.

I referred to this link:

[https://www.cs.toronto.edu/~lczhang/aps360\\_20191/lec/w03/convnet.html](https://www.cs.toronto.edu/~lczhang/aps360_20191/lec/w03/convnet.html)

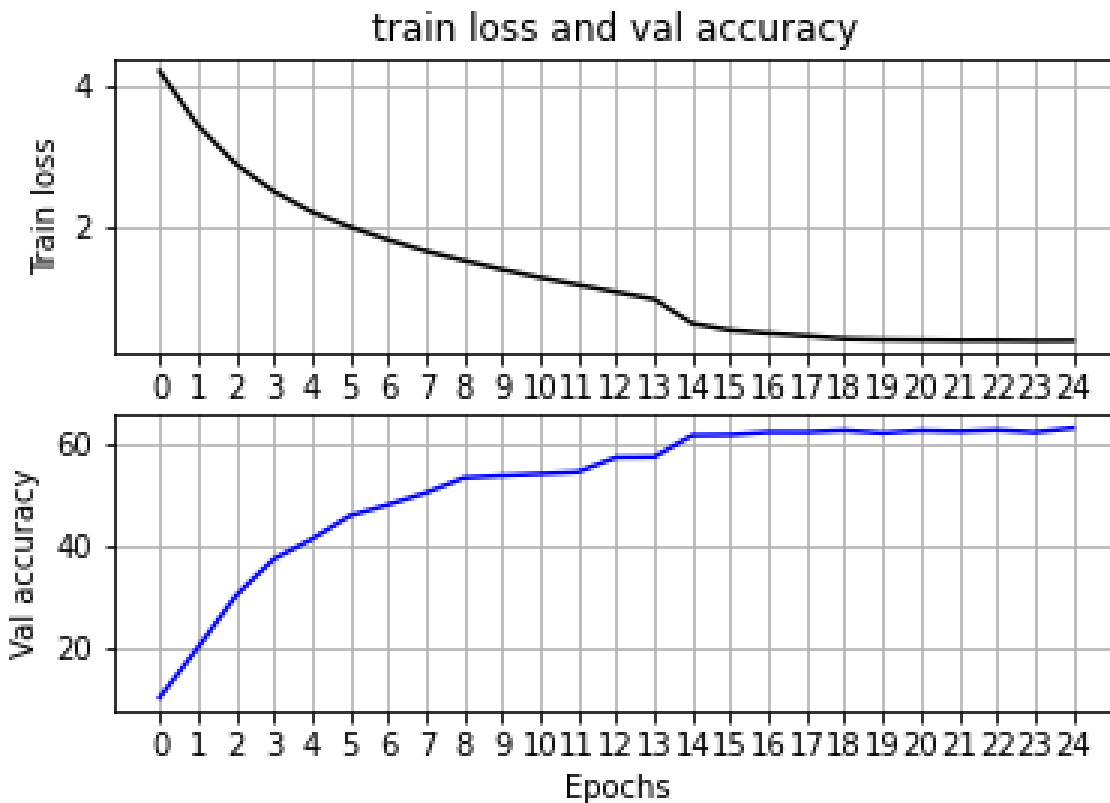
**The structure of final network is described in the table below:**

Layer No.	Layer Type	Kernel size (for conv layers)	Input   Output dimension	Input   Output Channels (for conv layers)
1	Conv2d	3	32 32	3 6
2	Batchnorm2d	-	32 32	-
3	Relu	-	32 32	-
4	Conv2d	3	32 32	6 16
5	Batchnorm2d	-	32 32	-
6	Relu	-	32 32	-
7	Conv2d	3	32 32	16 32
8	Batchnorm2d	-	32 32	-
9	Relu	-	32 32	-
10	Maxpool2d	2	32 16	-
11	Conv2d	3	16 16	32 64
12	Batchnorm2d	-	16 16	-
13	Relu	-	16 16	-
14	Conv2d	3	16 16	64 192
15	Batchnorm2d	-	16 16	-
16	Relu	-	16 16	-
17	Conv2d	3	16 16	192 384

18	Batchnorm2d	-	16 16	-
19	Relu	-	16 16	-
20	Maxpool2d	2	16 8	-
21	Conv2d	3	8 8	384 256
22	Batchnorm2d	-	8 8	-
23	Relu	-	8 8	-
24	Conv2d	3	8 8	256 256
25	Batchnorm2d	-	8 8	-
26	Relu	-	8 8	-
27	Maxpool2d	2	8 4	-
28	Linear	-	4096 2048	-
29	Relu	-	2048 2048	-
30	Linear	-	2048 1024	-
31	Relu	-	1024 1024	-
32	Dropout	-	1024 1024	-
33	Linear	-	1024 512	-
34	Relu	-	512 512	-
35	Linear	-	512 256	-
36	Relu	-	256 256	-
37	Liner	-	256 100	-

- The data was transformed to make it normal with zero mean and one standard deviation. The data was transformed with different parameters such as RandomHorizontalFlip() to make it more robust.
- After the transformation, the different convolution layers were added to make the network more deeper and to improve the training accuracy. After the convolutional layer, many fully connected layers were added.
- With all these changes, the accuracy also improved when the number of epochs were changed from 15 to 25.
- After changing epochs, different learning rate were also tested to improve the network and best outcome came with learning rate of 0.0035 and I applied scheduler as well which reduces learning rate after certain epochs.

**Plot illustrating the training loss and the validation accuracy:**

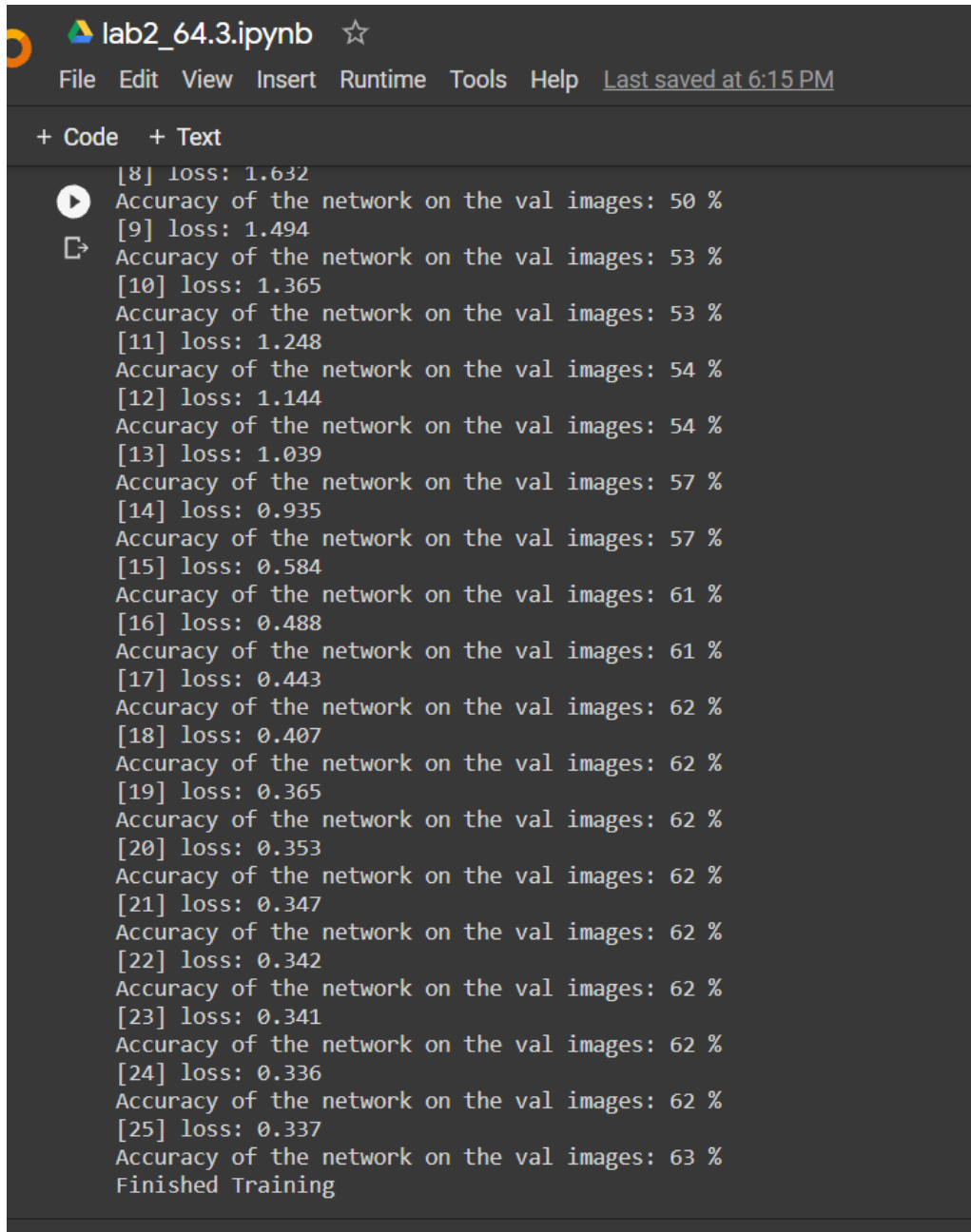


### **Ablation study:**

With the BaseNet that was mentioned earlier, the maximum accuracy achieved was around 26%. After adding convolution layer and changing number of epochs, the achieved accuracy was 51%. However, after trying many variations of different epochs and convolutional layer parameters, around 63% accuracy was achieved when learning rate was changed. The learning rate was changed from 0.005 to 0.0035 which improved the accuracy. Additionally, the learning rate is decreased more after certain epochs by using scheduler.

## Base performance:

The screenshot of the final accuracy of the network on the validation image is mentioned below:



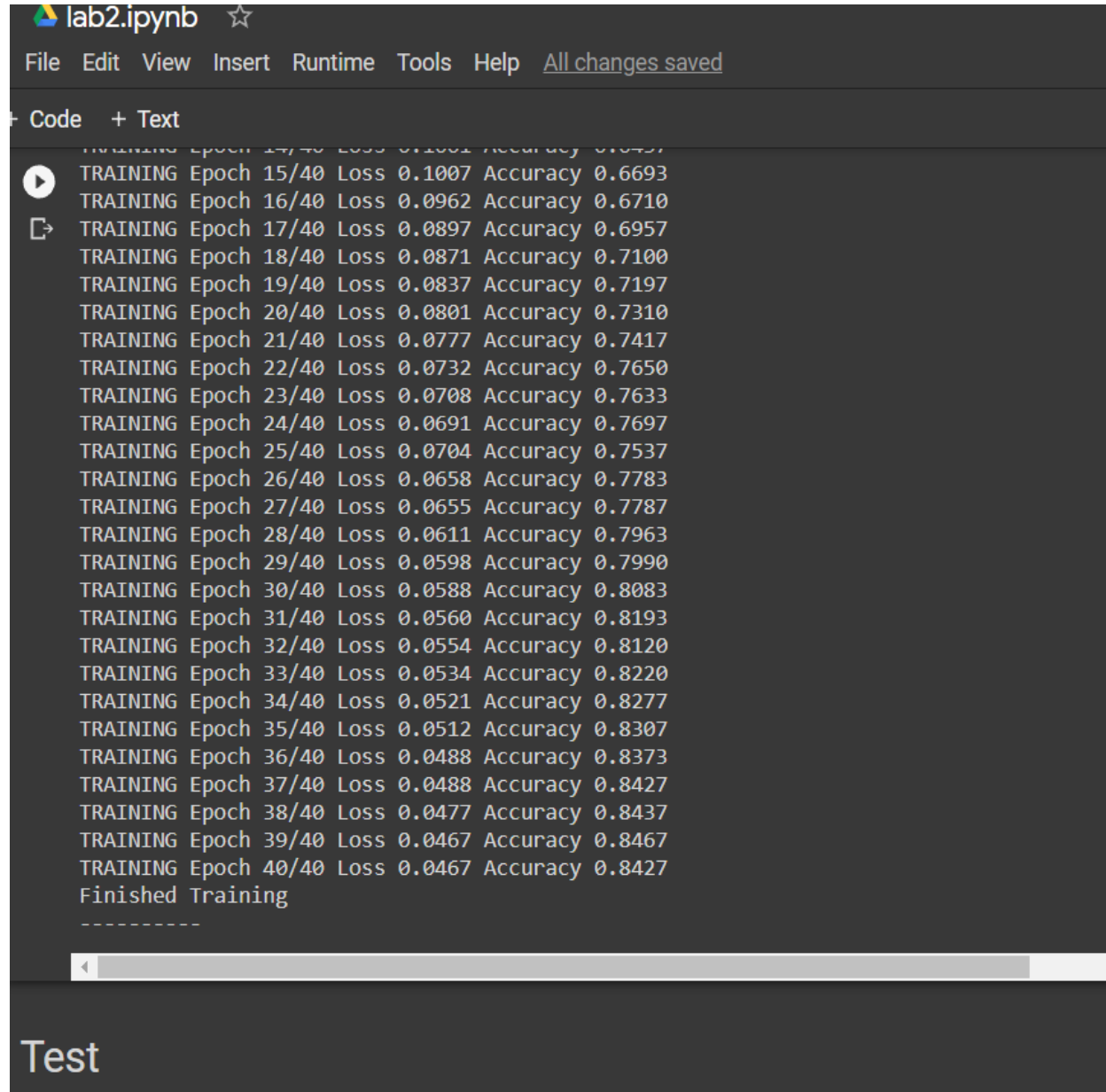
```
lab2_64.3.ipynb ☆
File Edit View Insert Runtime Tools Help Last saved at 6:15 PM
+ Code + Text
[8] loss: 1.632
Accuracy of the network on the val images: 50 %
[9] loss: 1.494
Accuracy of the network on the val images: 53 %
[10] loss: 1.365
Accuracy of the network on the val images: 53 %
[11] loss: 1.248
Accuracy of the network on the val images: 54 %
[12] loss: 1.144
Accuracy of the network on the val images: 54 %
[13] loss: 1.039
Accuracy of the network on the val images: 57 %
[14] loss: 0.935
Accuracy of the network on the val images: 57 %
[15] loss: 0.584
Accuracy of the network on the val images: 61 %
[16] loss: 0.488
Accuracy of the network on the val images: 61 %
[17] loss: 0.443
Accuracy of the network on the val images: 62 %
[18] loss: 0.407
Accuracy of the network on the val images: 62 %
[19] loss: 0.365
Accuracy of the network on the val images: 62 %
[20] loss: 0.353
Accuracy of the network on the val images: 62 %
[21] loss: 0.347
Accuracy of the network on the val images: 62 %
[22] loss: 0.342
Accuracy of the network on the val images: 62 %
[23] loss: 0.341
Accuracy of the network on the val images: 62 %
[24] loss: 0.336
Accuracy of the network on the val images: 62 %
[25] loss: 0.337
Accuracy of the network on the val images: 63 %
Finished Training
```

## Relative performance:

The csv file is submitted on Kaggle under the name RitikaGoyal with the score 0.64300.

## Part2: Transfer Learning

The screenshot of training accuracy is given below:



```
lab2.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
TRAINING Epoch 14/40 Loss 0.1001 Accuracy 0.6707
TRAINING Epoch 15/40 Loss 0.1007 Accuracy 0.6693
TRAINING Epoch 16/40 Loss 0.0962 Accuracy 0.6710
TRAINING Epoch 17/40 Loss 0.0897 Accuracy 0.6957
TRAINING Epoch 18/40 Loss 0.0871 Accuracy 0.7100
TRAINING Epoch 19/40 Loss 0.0837 Accuracy 0.7197
TRAINING Epoch 20/40 Loss 0.0801 Accuracy 0.7310
TRAINING Epoch 21/40 Loss 0.0777 Accuracy 0.7417
TRAINING Epoch 22/40 Loss 0.0732 Accuracy 0.7650
TRAINING Epoch 23/40 Loss 0.0708 Accuracy 0.7633
TRAINING Epoch 24/40 Loss 0.0691 Accuracy 0.7697
TRAINING Epoch 25/40 Loss 0.0704 Accuracy 0.7537
TRAINING Epoch 26/40 Loss 0.0658 Accuracy 0.7783
TRAINING Epoch 27/40 Loss 0.0655 Accuracy 0.7787
TRAINING Epoch 28/40 Loss 0.0611 Accuracy 0.7963
TRAINING Epoch 29/40 Loss 0.0598 Accuracy 0.7990
TRAINING Epoch 30/40 Loss 0.0588 Accuracy 0.8083
TRAINING Epoch 31/40 Loss 0.0560 Accuracy 0.8193
TRAINING Epoch 32/40 Loss 0.0554 Accuracy 0.8120
TRAINING Epoch 33/40 Loss 0.0534 Accuracy 0.8220
TRAINING Epoch 34/40 Loss 0.0521 Accuracy 0.8277
TRAINING Epoch 35/40 Loss 0.0512 Accuracy 0.8307
TRAINING Epoch 36/40 Loss 0.0488 Accuracy 0.8373
TRAINING Epoch 37/40 Loss 0.0488 Accuracy 0.8427
TRAINING Epoch 38/40 Loss 0.0477 Accuracy 0.8437
TRAINING Epoch 39/40 Loss 0.0467 Accuracy 0.8467
TRAINING Epoch 40/40 Loss 0.0467 Accuracy 0.8427
Finished Training
-----
Test
```

The screenshot of test accuracy is given below:

```
loss = criterion(outputs, labels)

_, preds = torch.max(outputs.data, 1)


test_loss += loss.item()
test_acc += torch.sum(preds == labels).item()

test_loss /= (dataset_sizes['test']*repeats)
test_acc /= (dataset_sizes['test']*repeats)

print('Test Loss: %.4f Test Accuracy %.4f' % (test_loss, test_acc))
```

```
[ ] test(model, criterion)
```

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader
  cpuset_checked))
Test Loss: 0.1017 Test Accuracy 0.5753
```



### Hyperparameter settings that were used:

- Number of epochs were changed to 40.
- Learning rate was changed to 0.001
- Batch size was increased to 16
- RESNET\_LAST\_ONLY was changed to false.