HLCV EXERCISE 3 REPORT

AKSHAY JOSHI
2581346
s8akjosh@stud.uni-saarland.de

ANKIT AGRAWAL
2581532
s8anagra@stud.uni-saarland.de

SUSHMITA NAIR
2581308
s8sunair@stud.uni-saarland.de

Question 1: Implement Convolutional Network

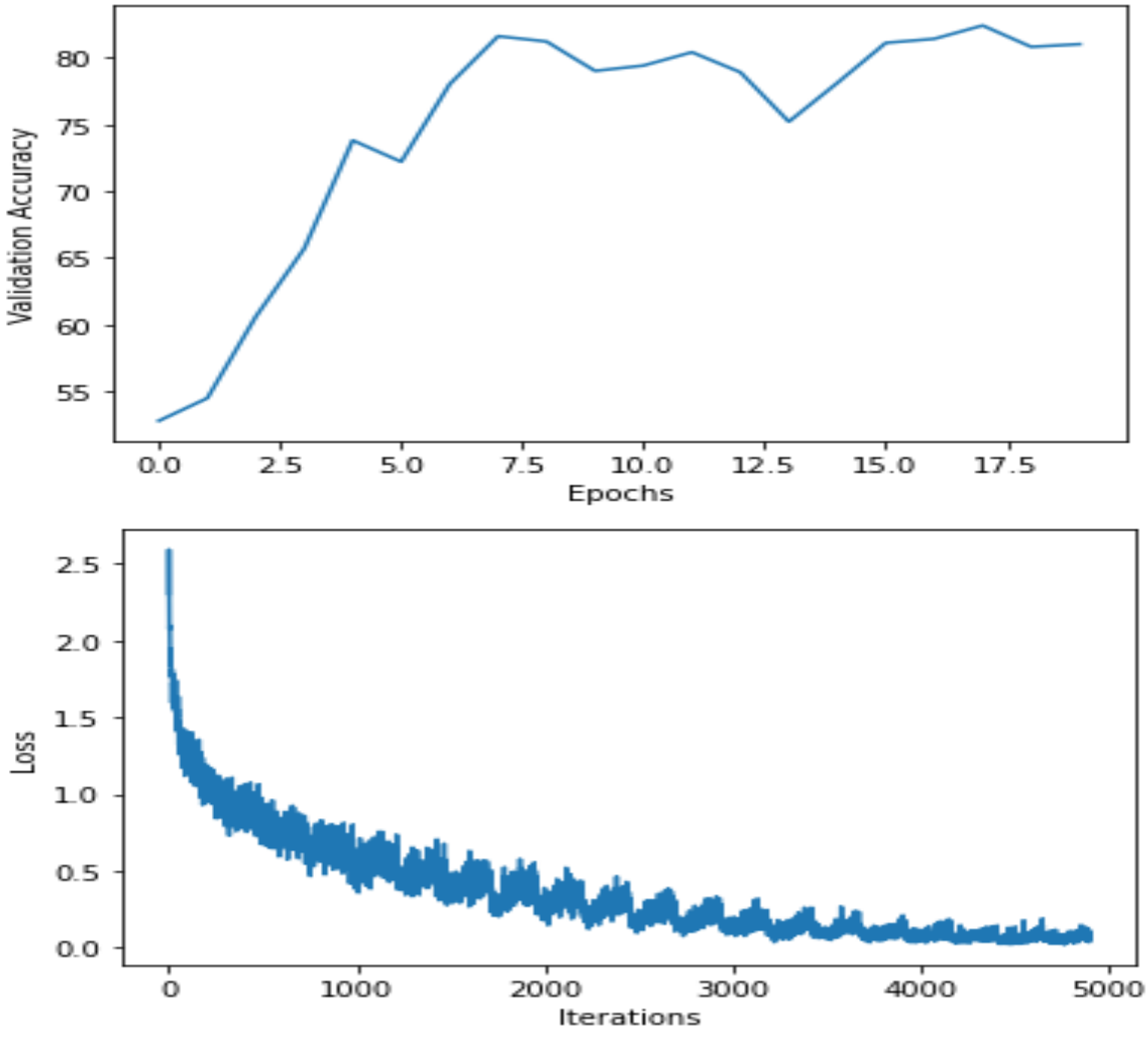
1. **Report the training and validation accuracies**

   We have used **Kaiming Weight Initialization** method considering the ReLu activation used in the network. Because of this we have achieved slightly better results compared to Random Weight Init.

   Validation Accuracy: **81.0 %**

   Accuracy of the network on the 1000 test images: **81.69999999999999 %**

   Plots to show the Accuracies & Loss of this model:

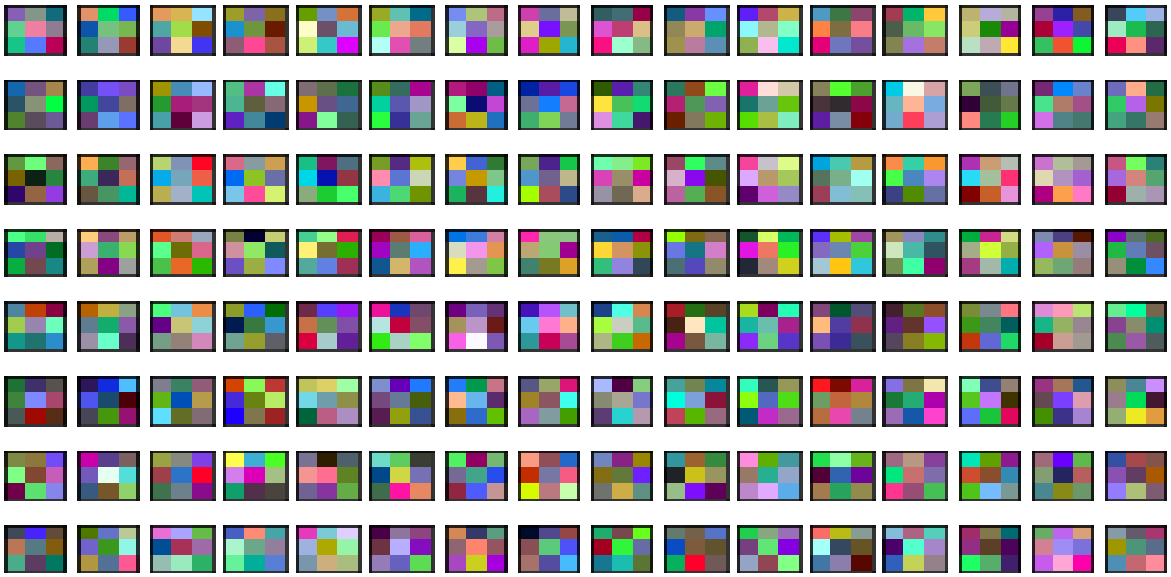


2. **Report the number of trainable parameters of the model**

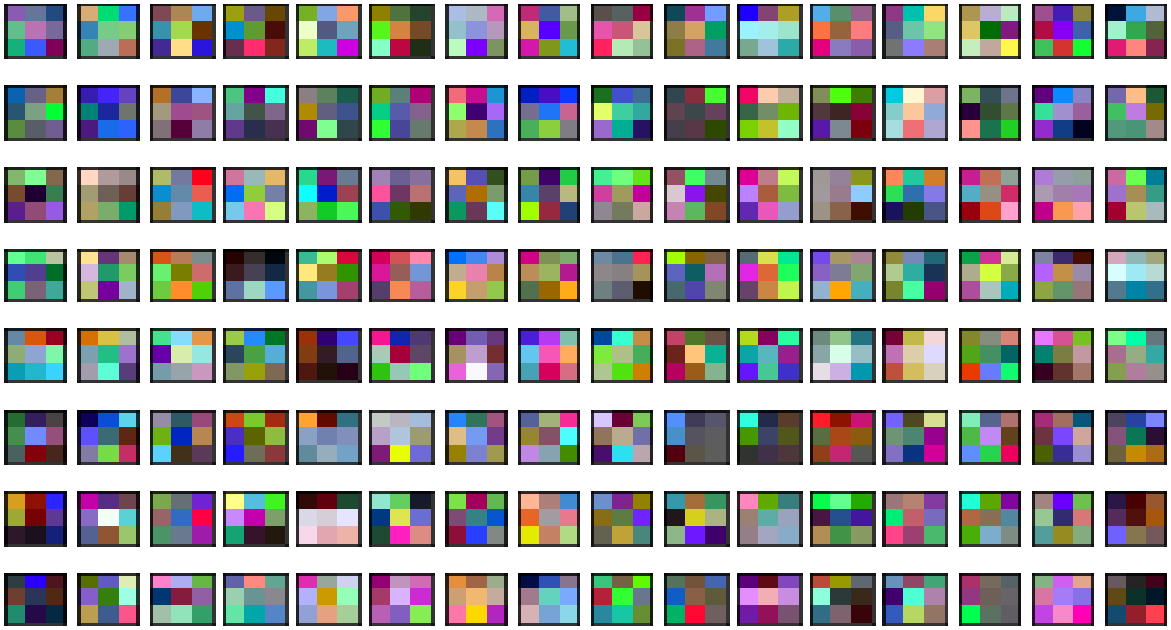
   Number of trainable parameters = 7682826

3. **Visualize & compare the filters before and after training**

Before:



After:



It can be noticed that the filters after the network training process have slightly changed based on certain conditions/regions of the image on which our kernel/filter passes through.

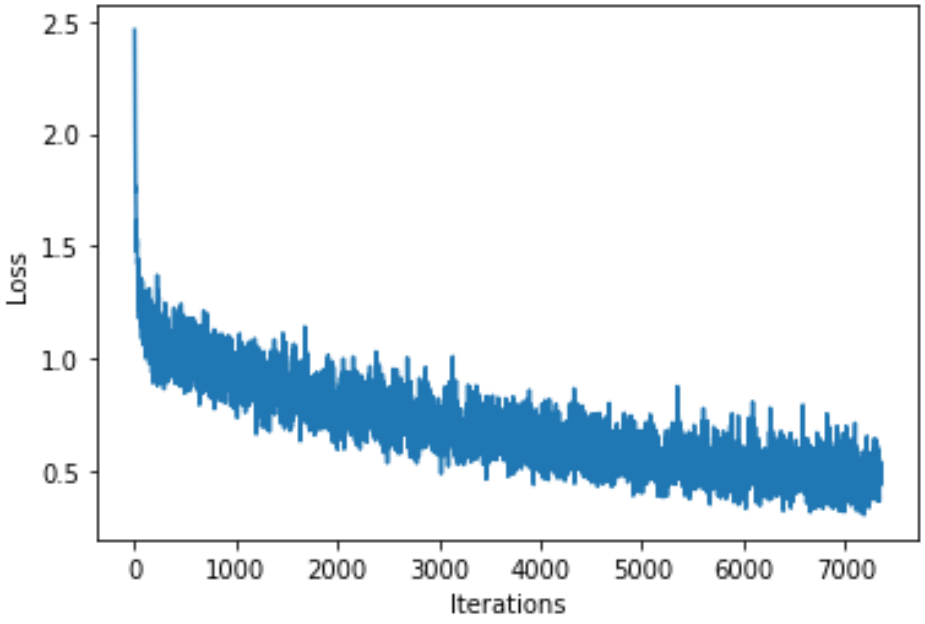
Some colors from the input are more pronounced and picked over other (Light intensity -> Dark in the output) and vice versa. Though it is sad that we can't

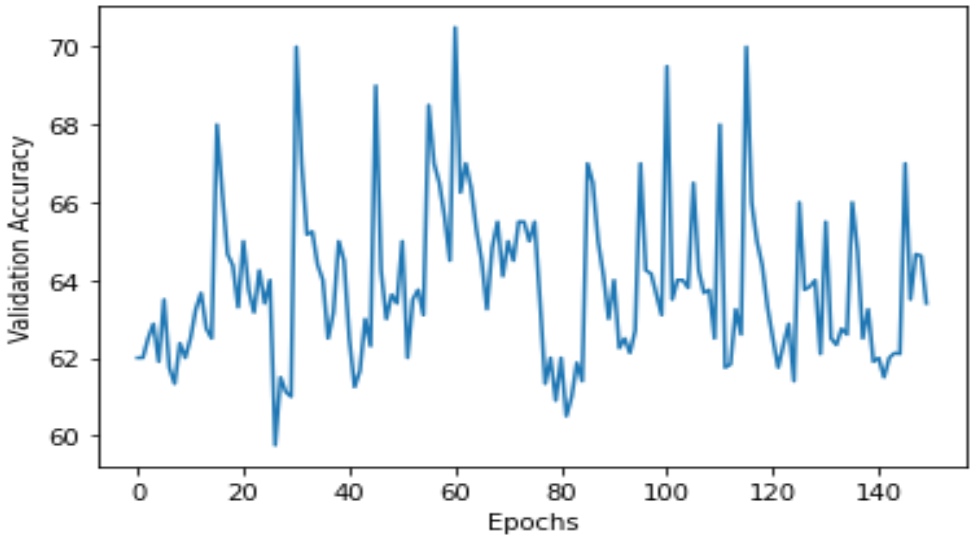
understand/see much about the edge detection or fine features detected by the network.

Question 4: Use pre-trained networks

1. **Validation and testing accuracies of Pre-trained Model:**

   Validation Accuracy: **64.8%**





2. Compare the two models training curves, validation and testing performance

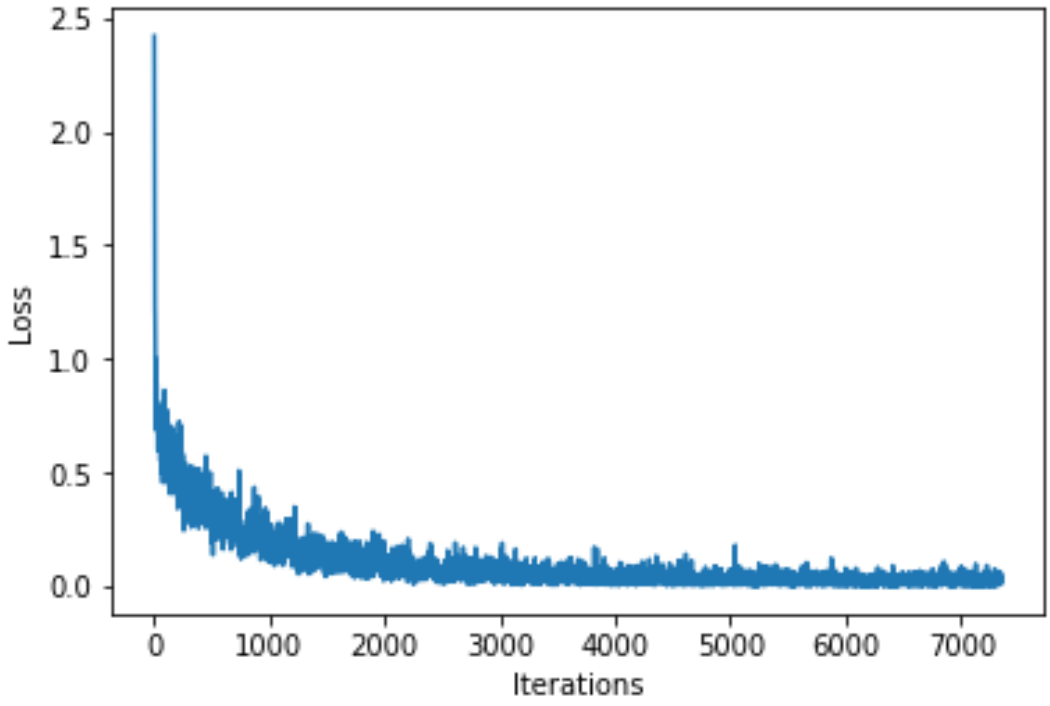
Please find the accuracy metrics for Model 2 (Fine-tuned) and Model 3 (Trained from scratch)

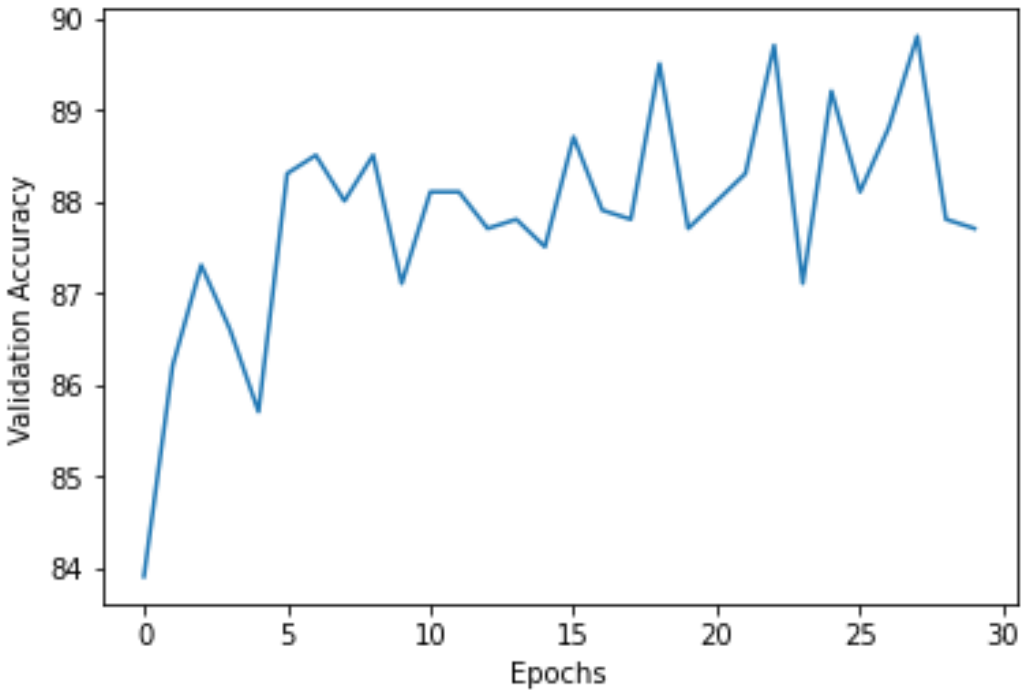
Test:
Accuracy of the best pre-trained network (using Early stopping) on the 1000 test images: **89.5 %**
Accuracy of the network with weights from random init on the 1000 test images: **88.9 %**

Validation:
Accuracy of the best pre-trained network (using Early stopping) on the 1000 test images: **89.9 %**
Accuracy of the network with weights from random init on the 1000 test images: **89.1 %**

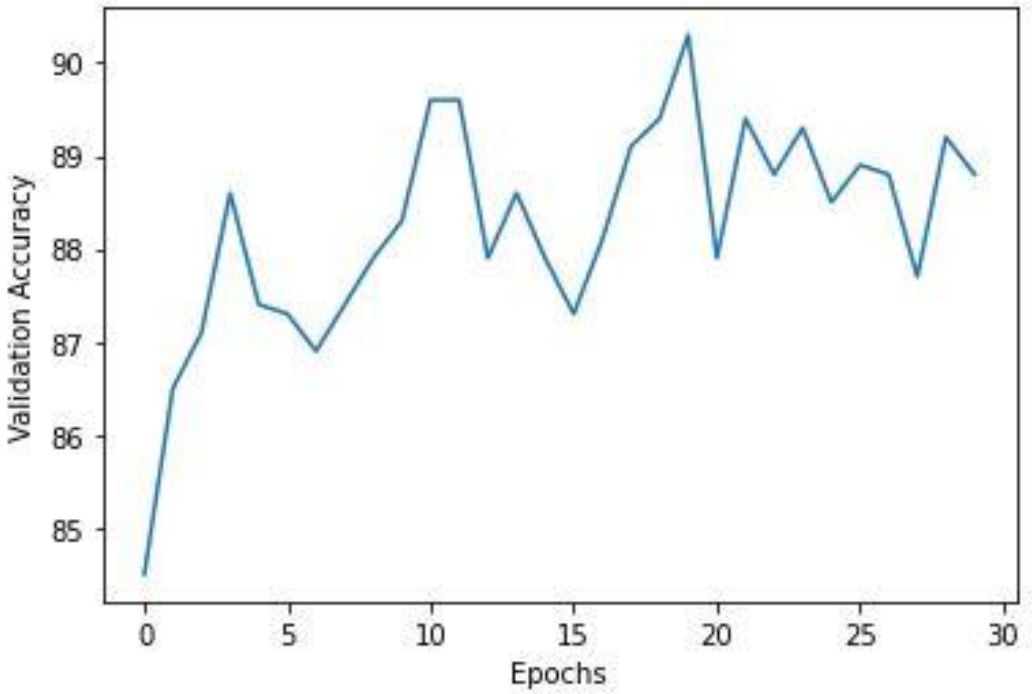
It appears that the fine-tuned model is slightly out-performing the model which was trained from scratch in both Test and Validation accuracies

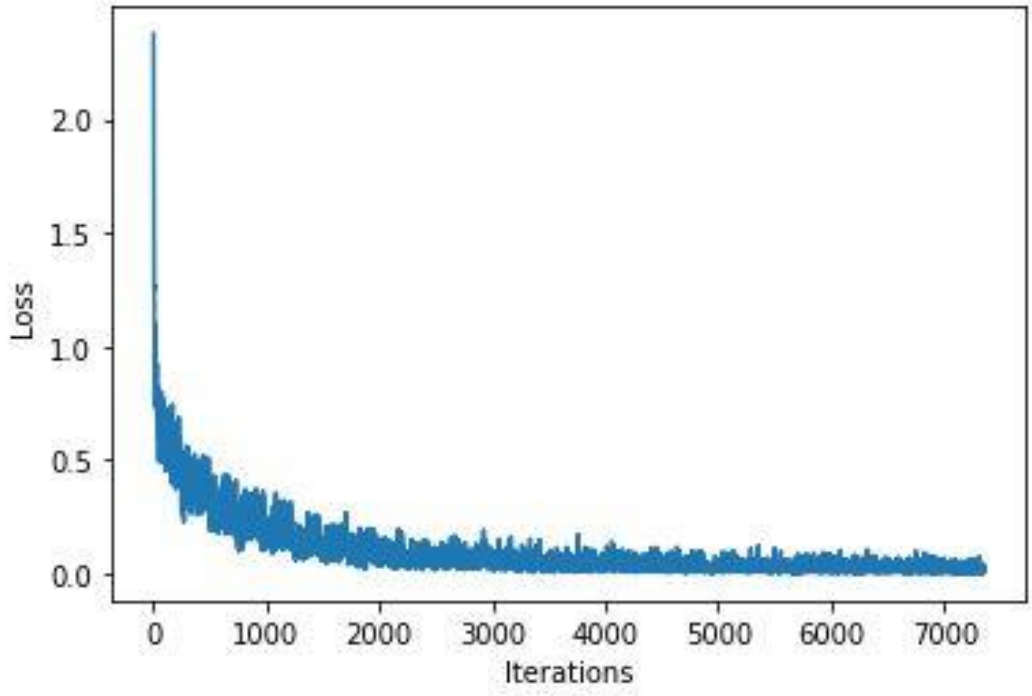
Model 2:
Test:




Validation Accuracy
90
89
88
87
86
85
84
0
5
10
15
20
25
30
Epochs

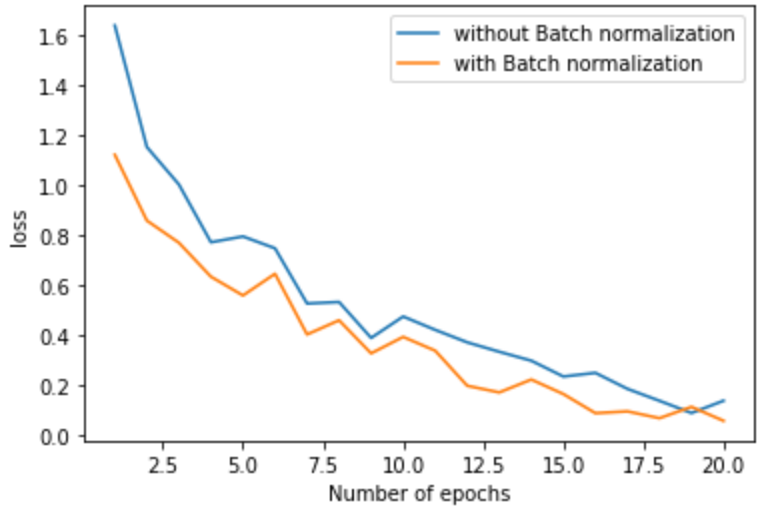
Val:


Validation Accuracy
90
89
88
87
86
85
0
5
10
15
20
25
30
Epochs


Loss
2.0
1.5
1.0
0.5
0.0
0
1000
2000
3000
4000
5000
6000
7000
Iterations

Question 2:

a)



Loss curve comparing loss without batch norm and with batch norm. We can see that with batch norm, the loss curve reaches lower values sooner indicating that faster train time is possible. The without batch norm loss curve is steeper and hence likely to take longer to converge. Also depending on the starting weights, the loss in initial epochs can be high, while with batch norm, the loss is smaller starting from the first epoch and also follows a less steep path to the minimum value. This also tells us that using batch norm makes the model robust to starting values(weights).
Batch norm also gives slightly better performance. without batch norm accuracy in validation set is 80.7% and with batch norm it is 81.8%

b)

Number of epochs was changed to 50. The best model with highest validation accuracy was obtained at epoch 35 as shown :

```
Epoch [35/50], Step [100/245], Loss: 0.0036
Epoch [35/50], Step [200/245], Loss: 0.0072
Validataion accuracy is: 84.9 %
Best
```

After this point, all epochs show lesser loss values but lower accuracy values. **This indicates overfitting.**

```
Epoch [36/50], Step [100/245], Loss: 0.0039
```

```
Epoch [36/50], Step [200/245], Loss: 0.0037
Validataion accuracy is: 82.9 %
Epoch [37/50], Step [100/245], Loss: 0.0049
Epoch [37/50], Step [200/245], Loss: 0.0055
Validataion accuracy is: 83.3 %

.
.
.

Epoch [50/50], Step [100/245], Loss: 0.0038
Epoch [50/50], Step [200/245], Loss: 0.0054
Validataion accuracy is: 83.0 %
Accuracy of the network on the 1000 test images: 83.8 %
```

The latest model from epoch 50 has lower validation accuracy of 83% and also lower test error of 81.9%
But the best model was found much earlier in epoch 35 with validation accuracy for 84.9% and test accuracy of 83.8%. Therefore early stopping can be used to alleviate overfitting.

3a) Data Augmentation

Experiment Setup:

Here we use different data transformations:
For Geometric augmentation, we use:

- Resize
- Crop
- HorizontalFlip
- Rotation

For color space data augmentation, we use:

- ColorJitter
- GrayScale

We check the performance of each transformation individually using some combinations of hyperparameters to get higher accuracy values.

```
input_size = 3
num_classes = 10
hidden_size = [128, 512, 512, 512, 512, 512]
num_epochs = 30 #Default: 20
batch_size = 200 #Default: 200
learning_rate = 2e-3 #Default: 2e-3
learning_rate_decay = 0.95 #Default: 0.95
reg=0.001 #Default: 0.001
num_training= 49000
num_validation = 1000
norm_layer = True ("batch")
use_dropout = False
```

Note: here we use epochs of size 30.

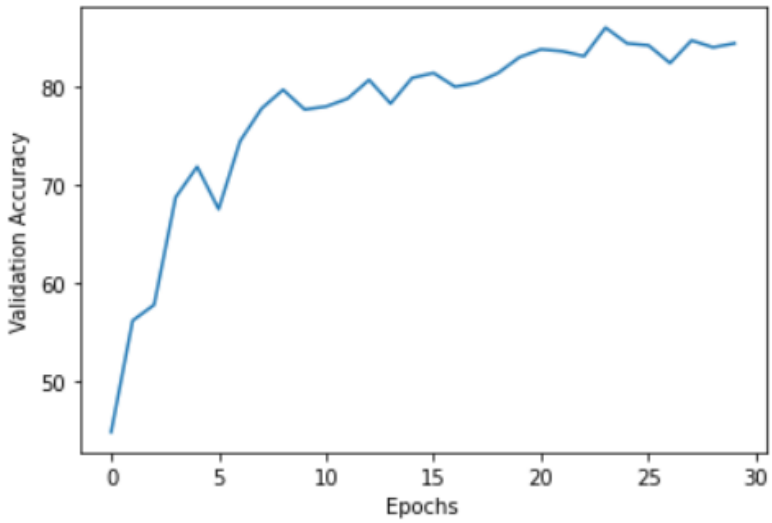
Results:

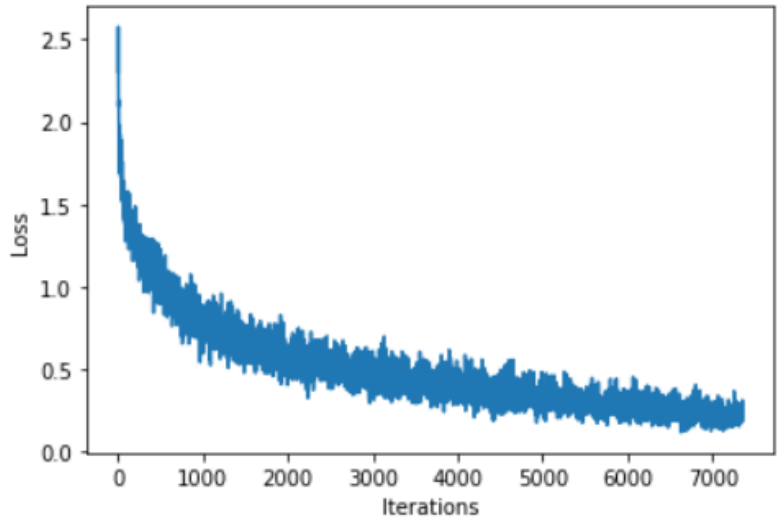
1. Using Resize and RandomCrop:

```
transforms.Resize((40, 40)),
transforms.RandomCrop((32, 32))
```

The original image is of size 32*32 so we first resize to 40*40 and then do a random crop to get image of 32*32.

We get.
Validation Accuracy: 85.9 %
Test Accuracy: 75.1 %



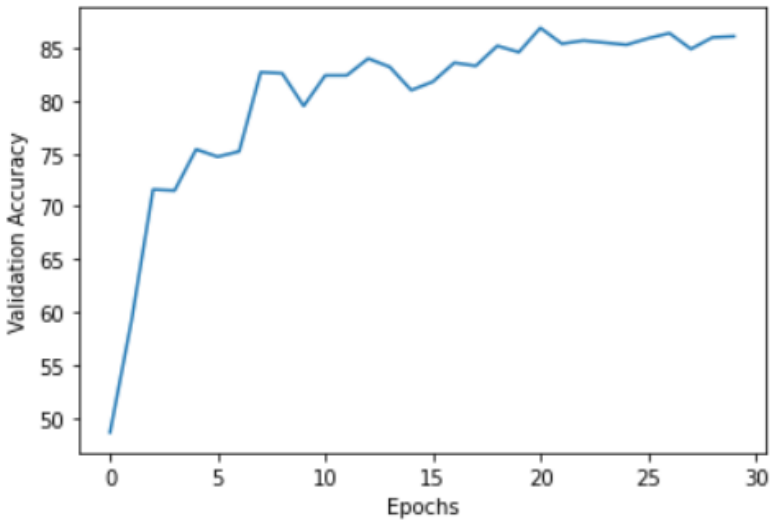


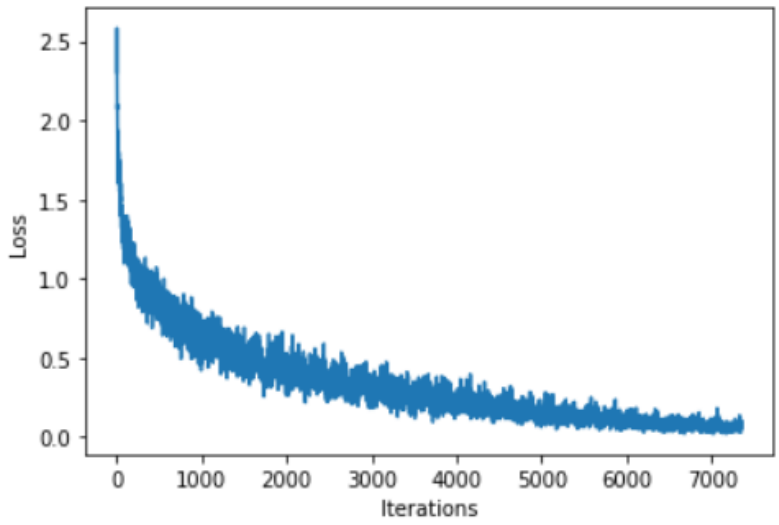
2. Using Horizontal Flip: (0.5)

```
transforms.RandomHorizontalFlip(),
```

Horizontally flip the given PIL Image randomly with a given probability.
Default = 0.5

We get.
Validation Accuracy: 86.9 %
Test Accuracy: 84.5 %



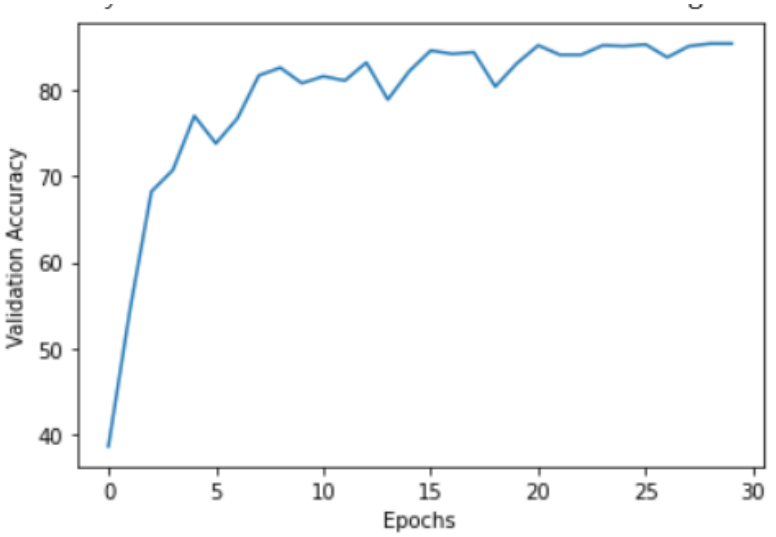


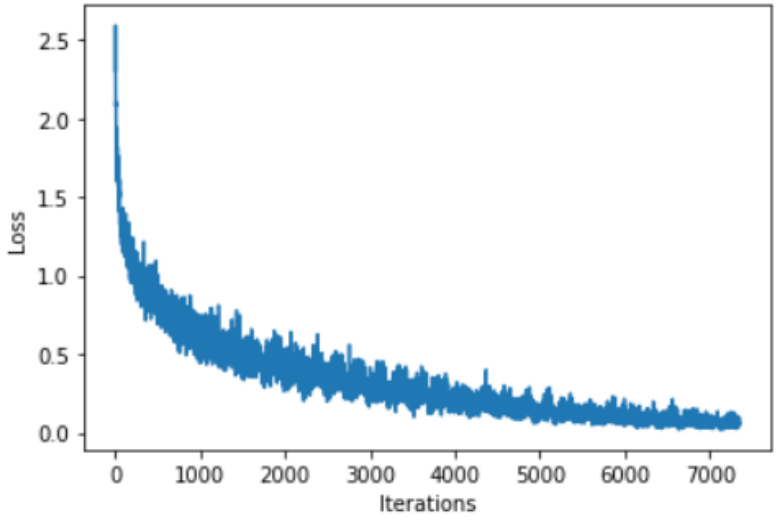
3. **Using Horizontal Flip:** (0.6)

```
transforms.RandomHorizontalFlip(0.6),
```

Horizontally flip the given PIL Image randomly with a given probability.
Default = 0.5

We get.
Validation Accuracy: 85.4 %
Test Accuracy: 86.0 %





4. Using Rotation: (degree = 15)

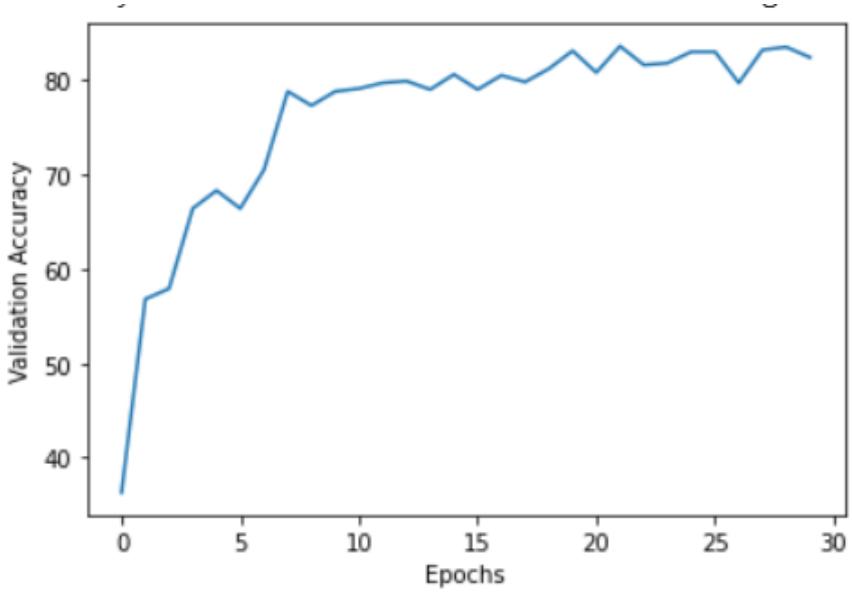
```
transforms.RandomRotation(degree = 15),
```

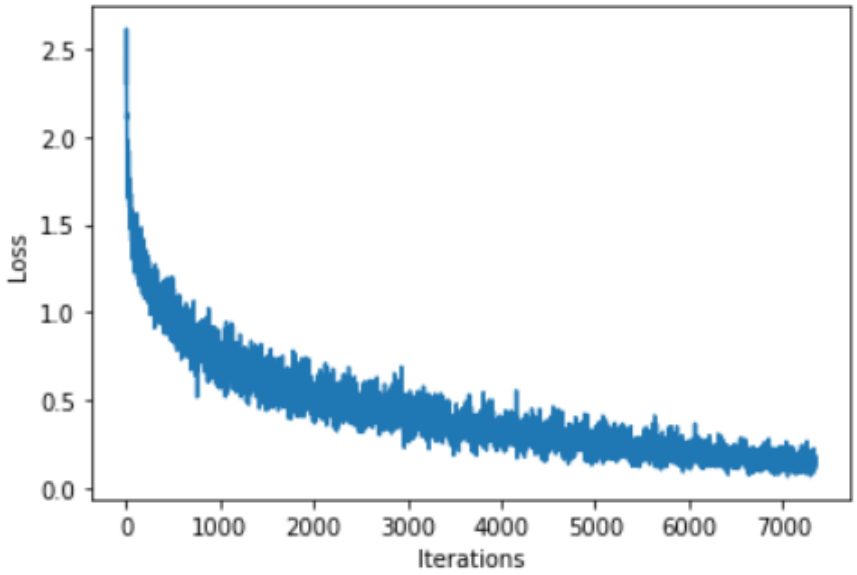
Rotate the image by angle. Here degree is the range of degree to select from, for above case it will be in the range of (-15, +15)

We get.

Validation Accuracy: 83.6 %

Test Accuracy: 83.3 %





5. Using Rotation: (degree = 25)

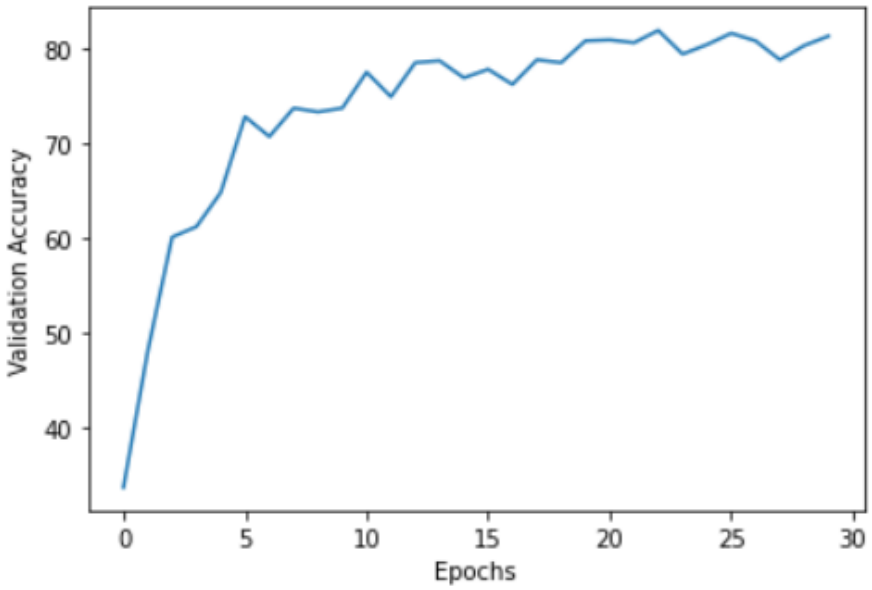
```
transforms.RandomRotation(degree = 25),
```

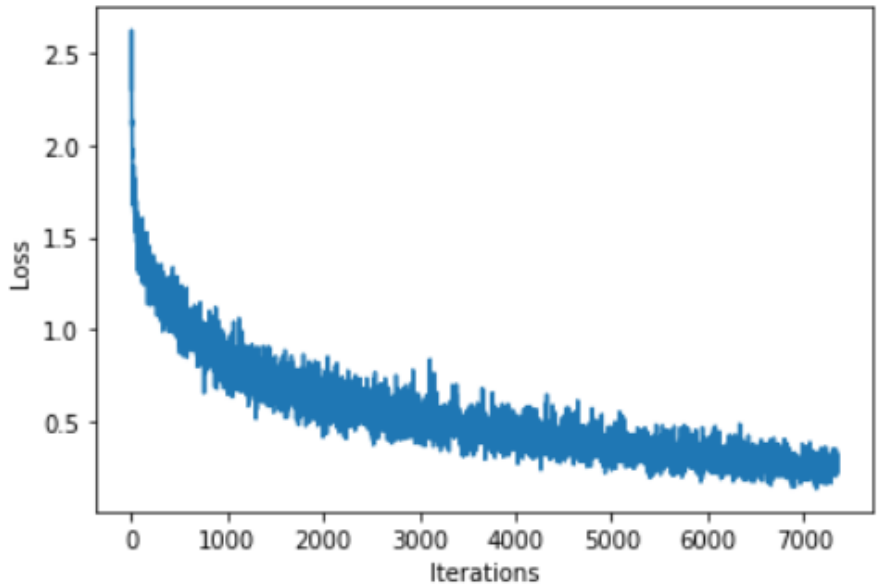
Rotate the image by angle. Here degree is the range of degree to select from, for above case it will be in the range of (-25, +25)

We get.

Validation Accuracy: 81.9 %

Test Accuracy: 83.3 %





6. Using colorJitter

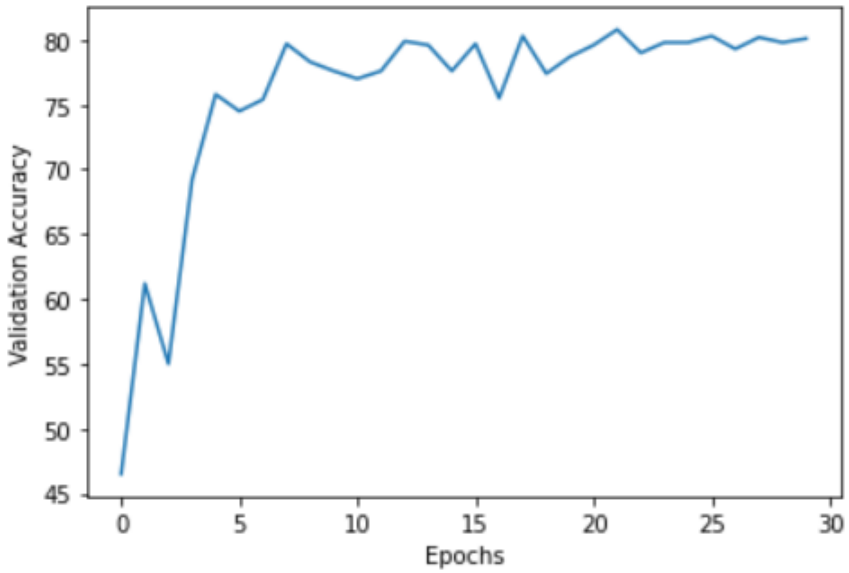
```
transforms.ColorJitter(brightness=0.4, contrast=0.4, saturation=0.4,
hue=0.1),
```

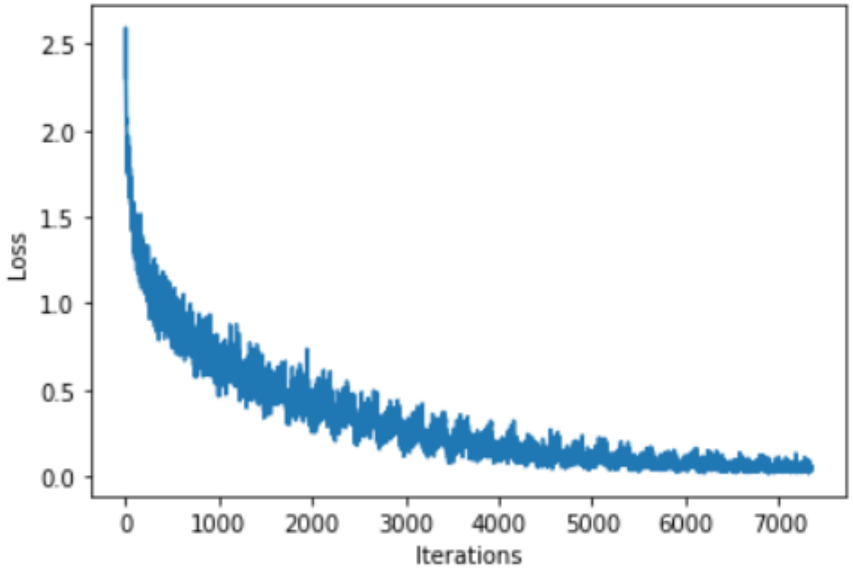
Randomly change the brightness, contrast, and saturation of an image.

We get.

Validation Accuracy: 80.8 %

Test Accuracy: 84.1 %





7. Using Grayscale:

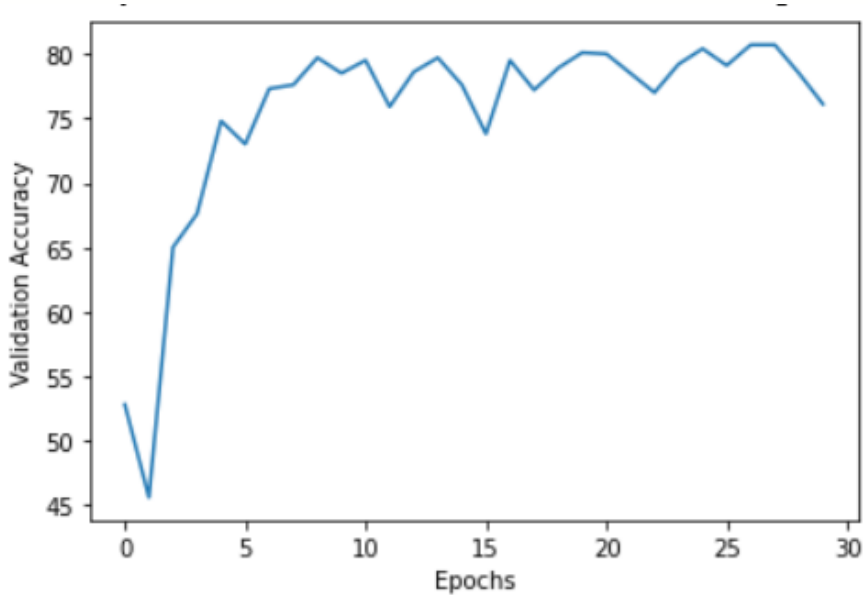
```
torchvision.transforms.Grayscale(num_output_channels=3)
```

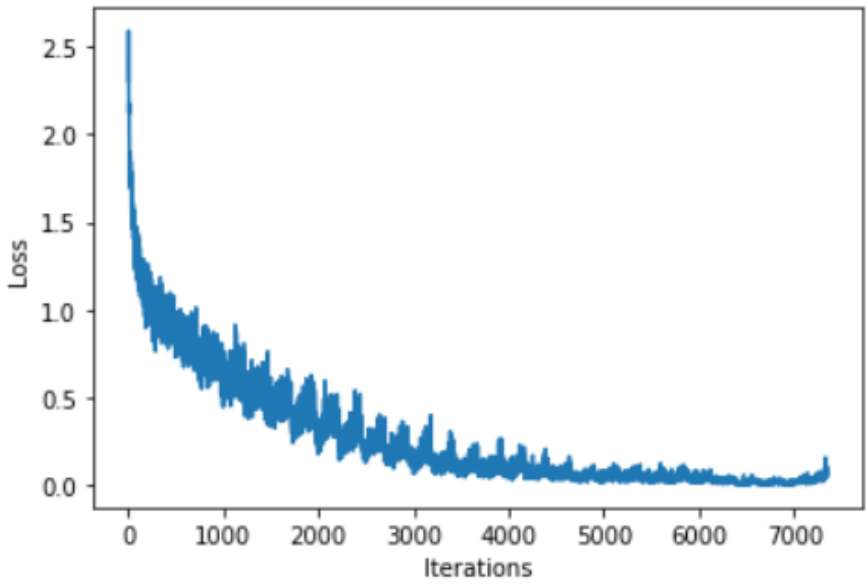
The original image is of size 32*32 so we first resize to 40*40 and then do a random crop to get image of 32*32.

We get.

Validation Accuracy: 80.4 %

Test Accuracy: 59.8 %





Observations:

We can see from using different data transformations:

The best accuracy is given by using HorizontalFlip(0.5) i.e. 86.9%

We also notice that **Geometric data transformation performs better than color space data augmentations.**

Among which GrayScale transformation performs the worst.

Question 3b (Dropout)

Experimental Setup :

```
input_size = 3
num_classes = 10
hidden_size = [128, 512, 512, 512, 512, 512]
num_epochs = 20 #Default: 20
batch_size = 200 #Default: 200
learning_rate = 2e-3 #Default: 2e-3
learning_rate_decay = 0.95 #Default: 0.95
reg=0.001 #Default: 0.001
dropout_value = [0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
num_training= 49000
num_validation =1000
norm_layer = True  # (Batch Normalization)
data_transform = False
```

We experiment for different values of p for dropout ranging from 0.1 to 0.9.

Results :

Test and validation accuracies have been reported for each value of p.

Model	p-value	Validation Accuracy	Test Accuracy
Model 1	0.1	84.40%	85.40%
Model 2	0.15	85.20%	84.10%
Model 3	0.2	85.80%	85.00%
Model 4	0.25	85.80%	84.30%
Model 5	0.3	85.80%	84.90%
Model 7	0.4	83.20%	81.10%
Model 8	0.5	79.90%	79.20%
Model 9	0.6	75.70%	76.10%
Model 10	0.7	68.10%	66.10%
Model 11	0.8	38.90%	37.00%
Model 12	0.9	11.20%	12.40%

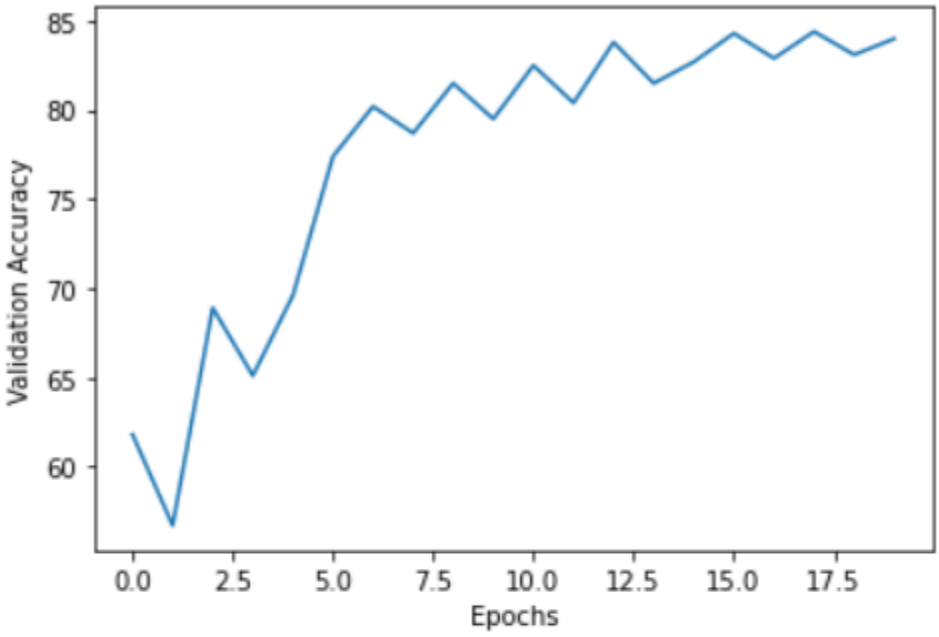
Observations:

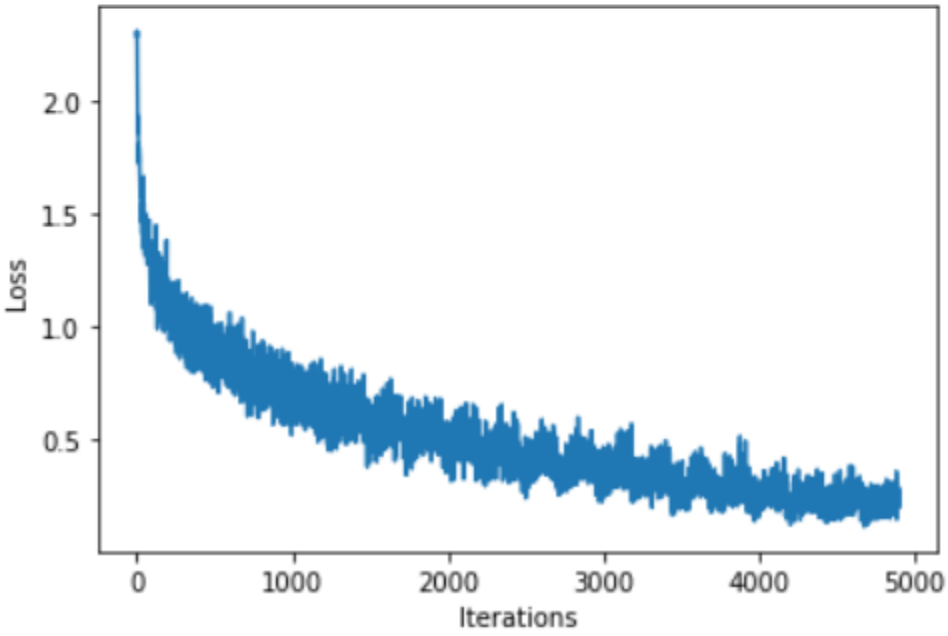
We get maximum validation accuracy for Model with Dropout probability(p) = 0.2
I.e. Validation Accuracy = 85.50%
And Test Accuracy = 85.00%

From the plots(below) we can also notice that high values of p regularize the model heavily and decrease model capacity, but with low values, the model overfits.

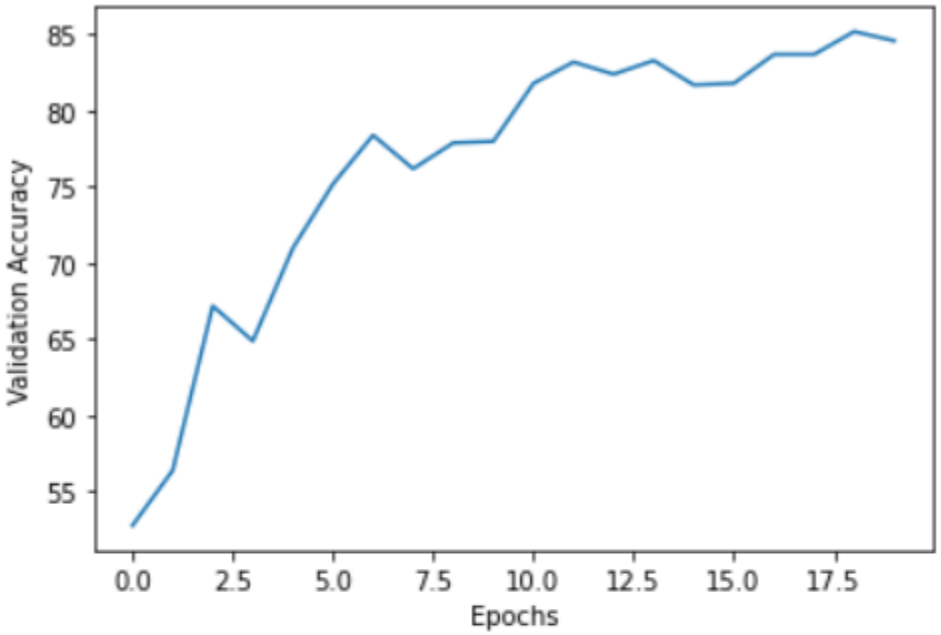
Plots for Accuracy and Loss for different values of p (Dropout probability):

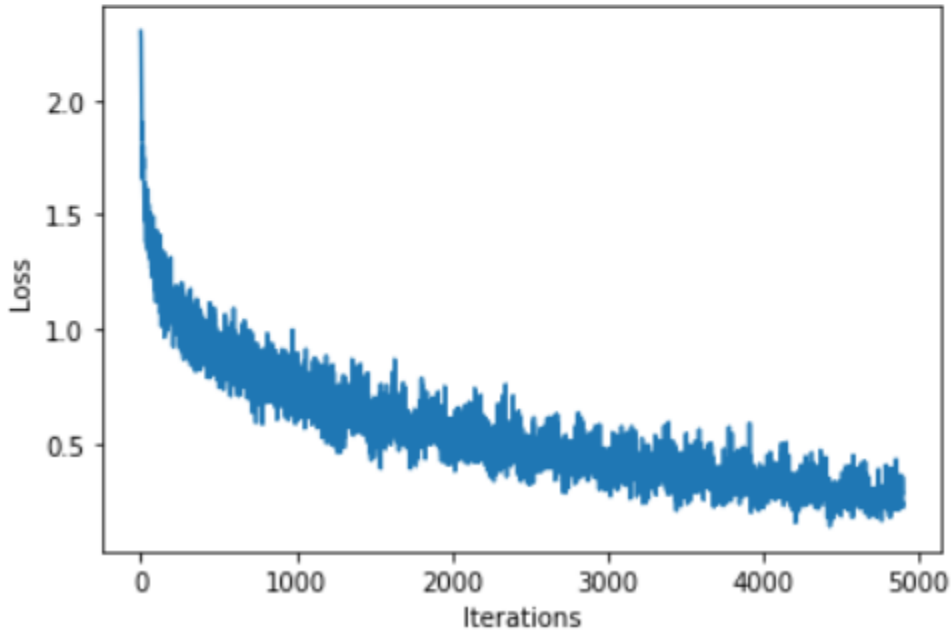
p = 0.1



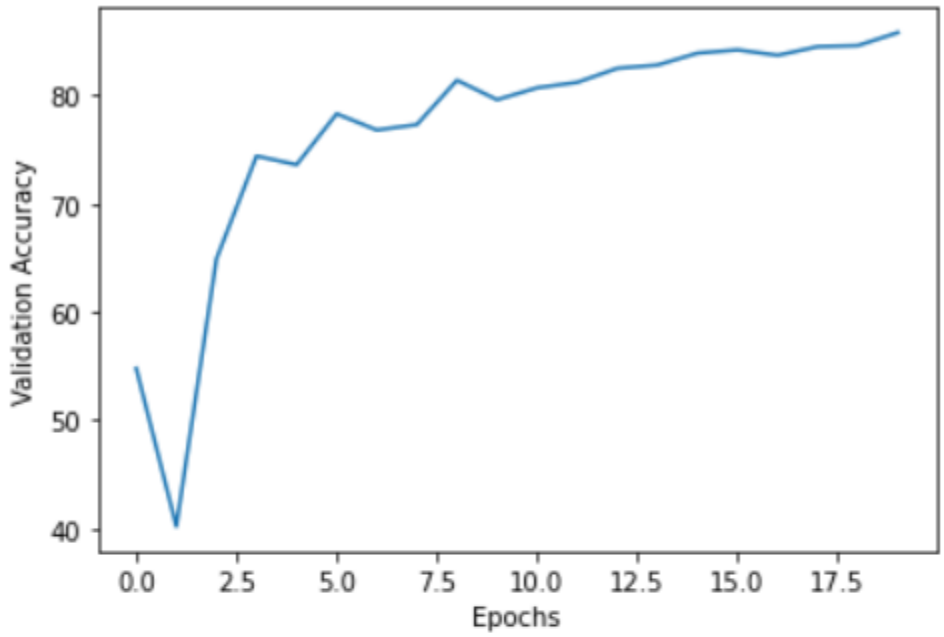


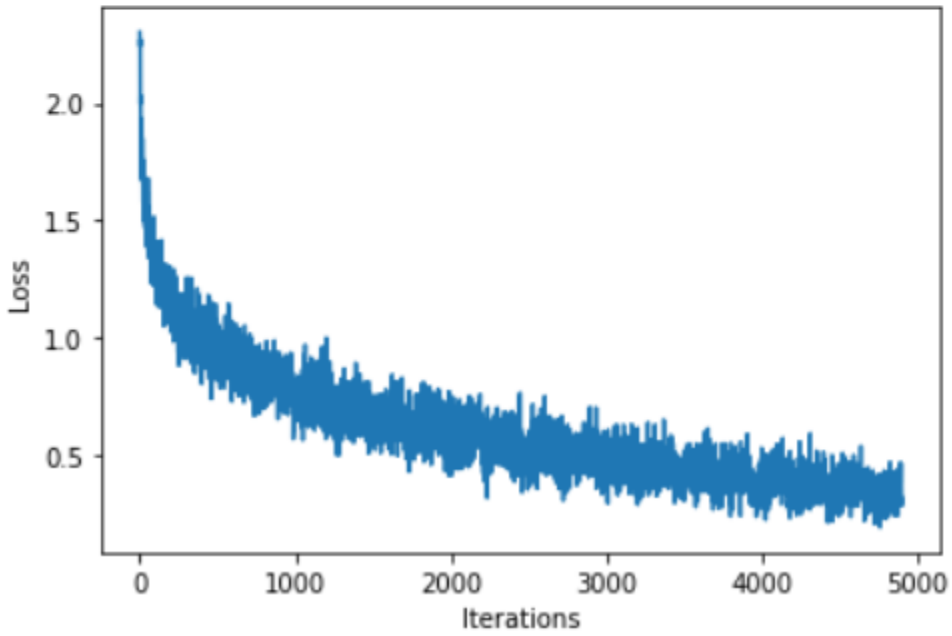
P = 0.15


Validation Accuracy
85
80
75
70
65
60
55
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs

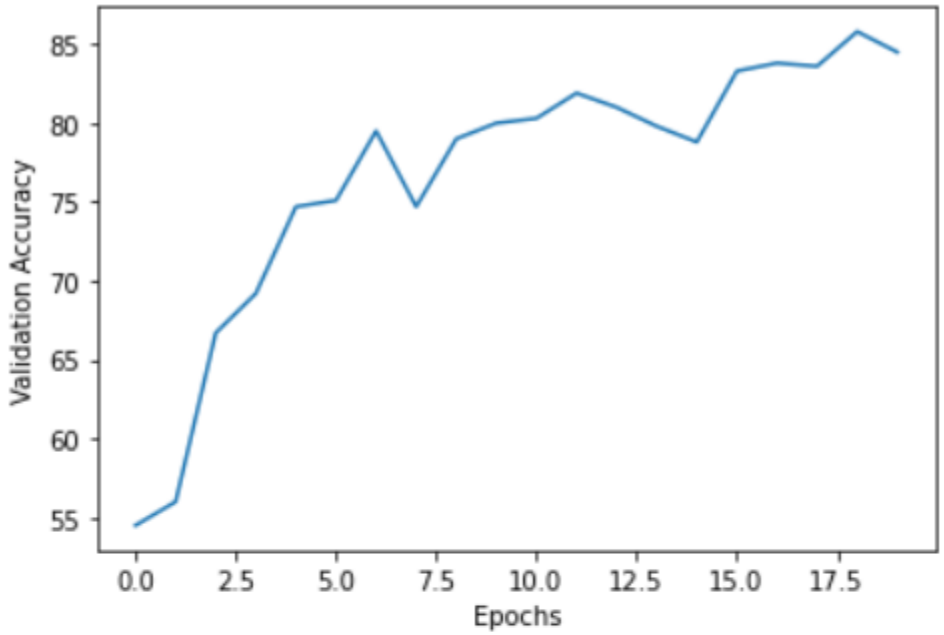

Loss
2.0
1.5
1.0
0.5
0
1000
2000
3000
4000
5000
Iterations

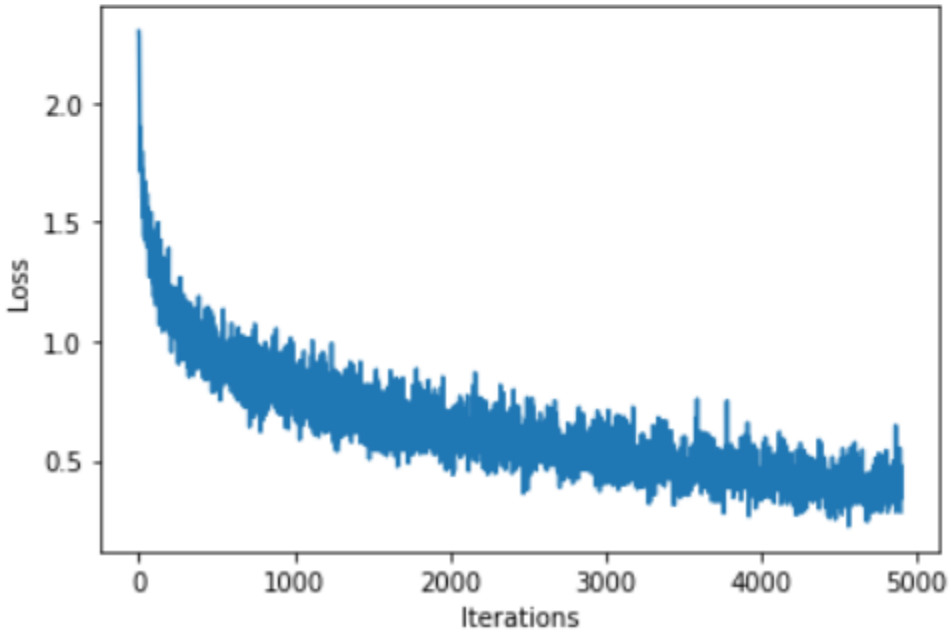
P = 0.2


Validation Accuracy
80
70
60
50
40
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs

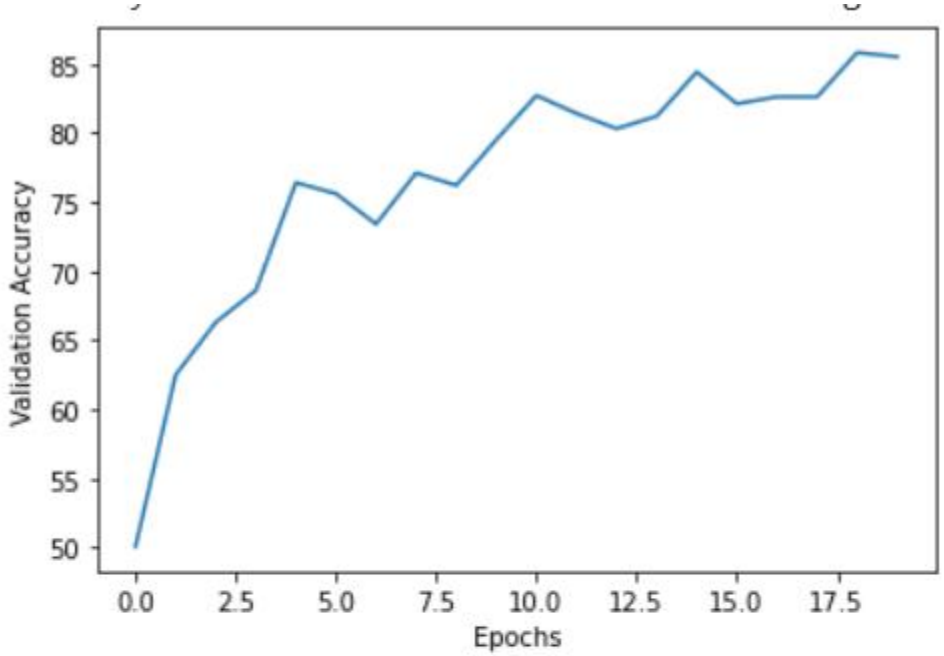

Loss
2.0
1.5
1.0
0.5
0
1000
2000
3000
4000
5000
Iterations

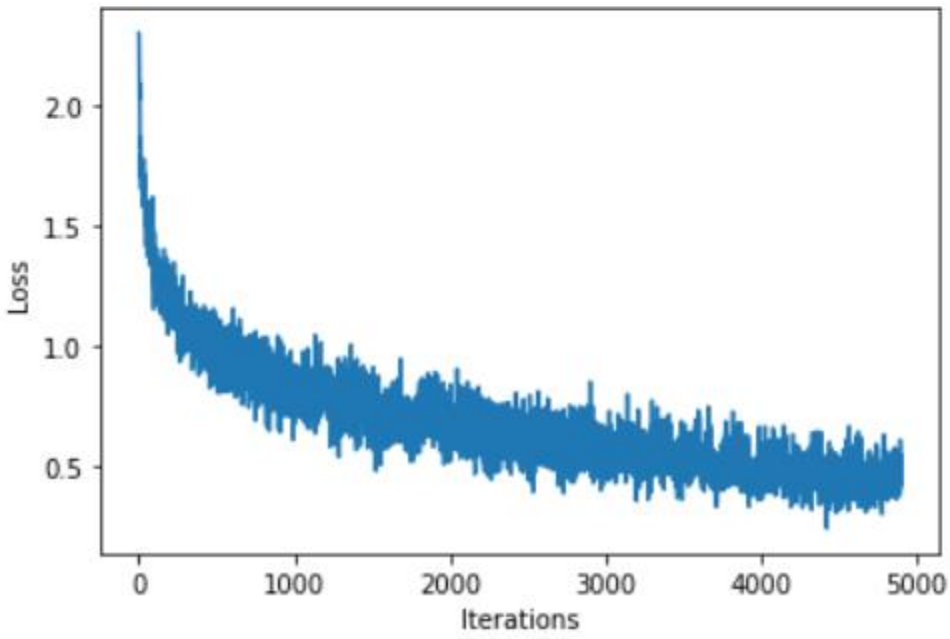
P = 0.25


Validation Accuracy
85
80
75
70
65
60
55
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs

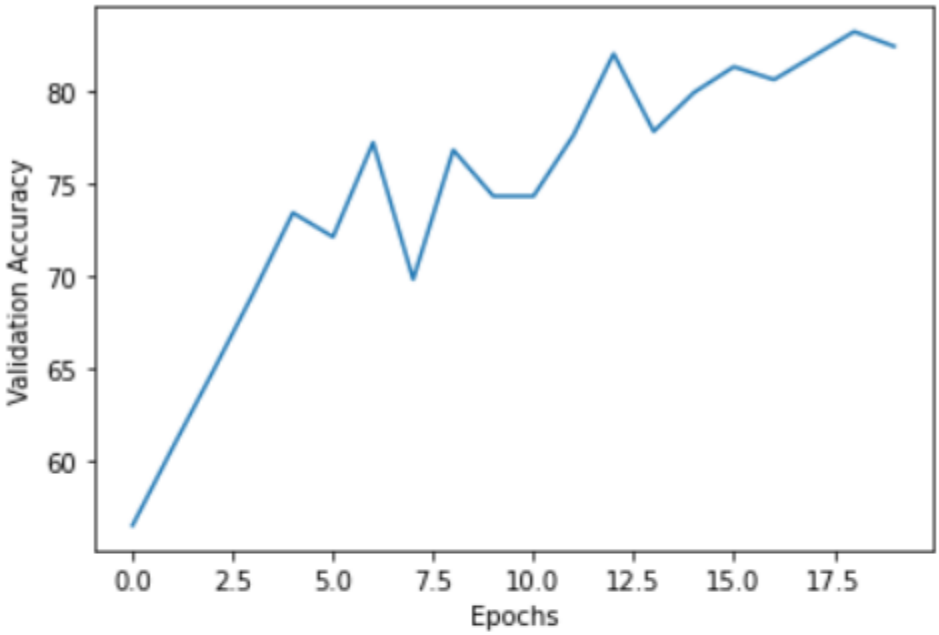

Loss
2.0
1.5
1.0
0.5
0
1000
2000
3000
4000
5000
Iterations

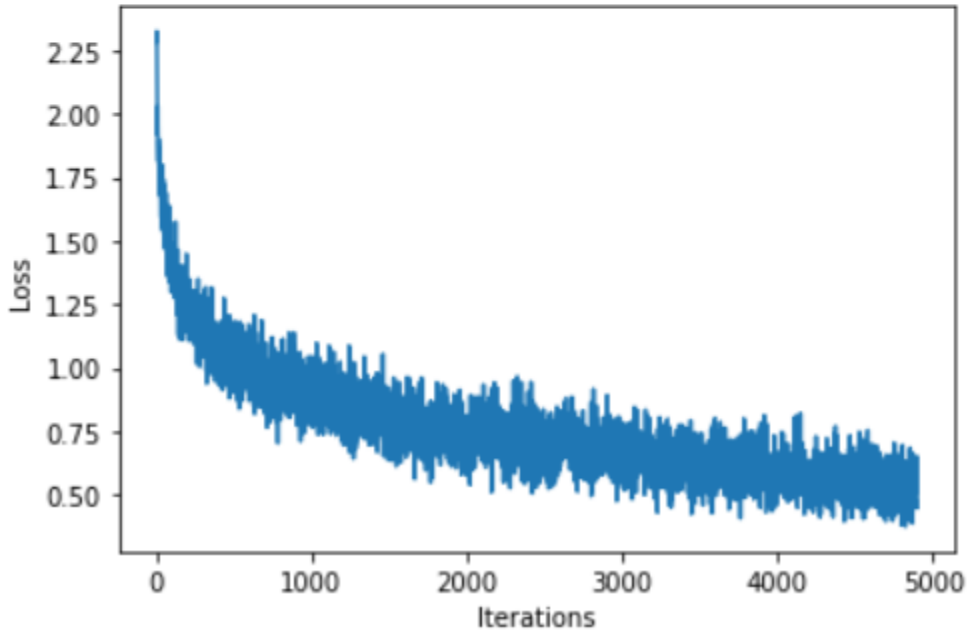
P = 0.3


Validation Accuracy
85
80
75
70
65
60
55
50
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs

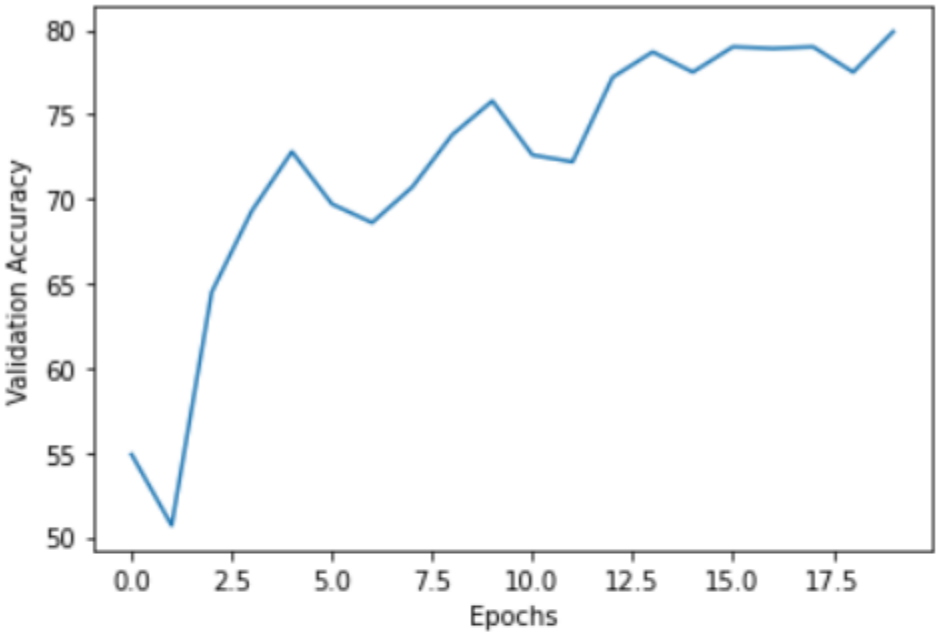

Loss
2.0
1.5
1.0
0.5
0
1000
2000
3000
4000
5000
Iterations

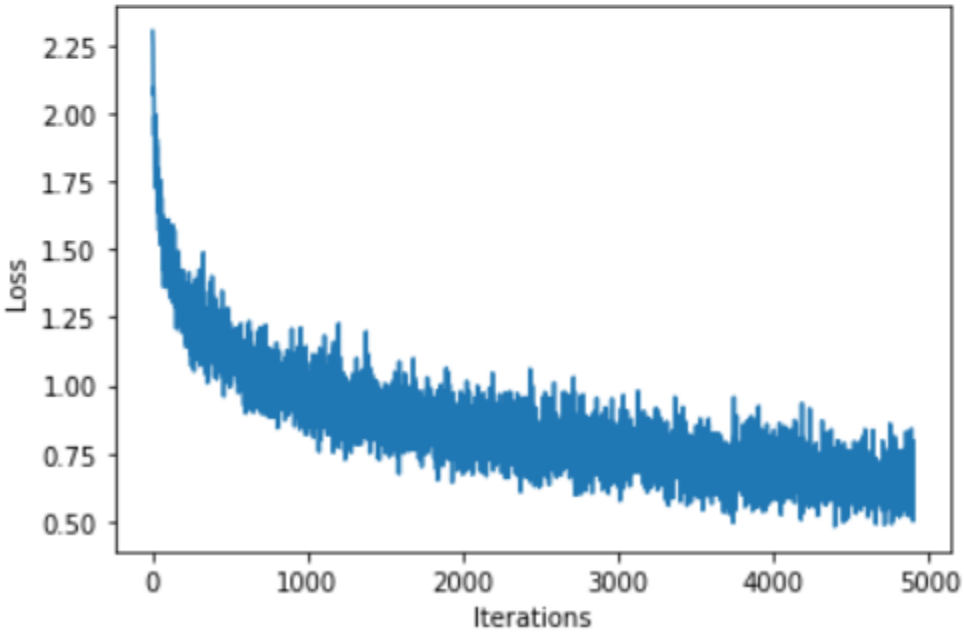
p = 0.4


Validation Accuracy
80
75
70
65
60
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs

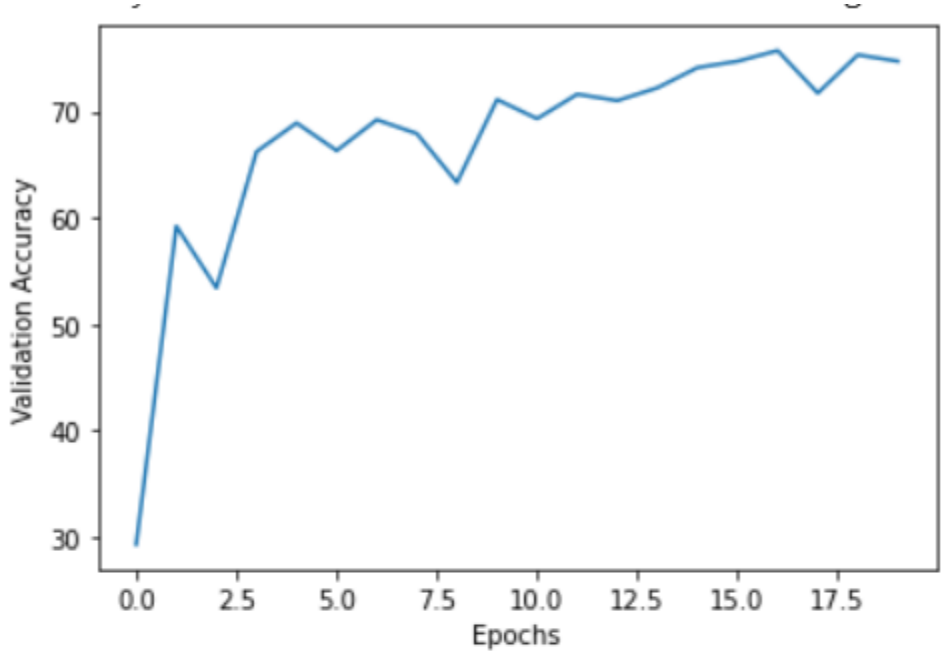

Loss
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0
1000
2000
3000
4000
5000
Iterations

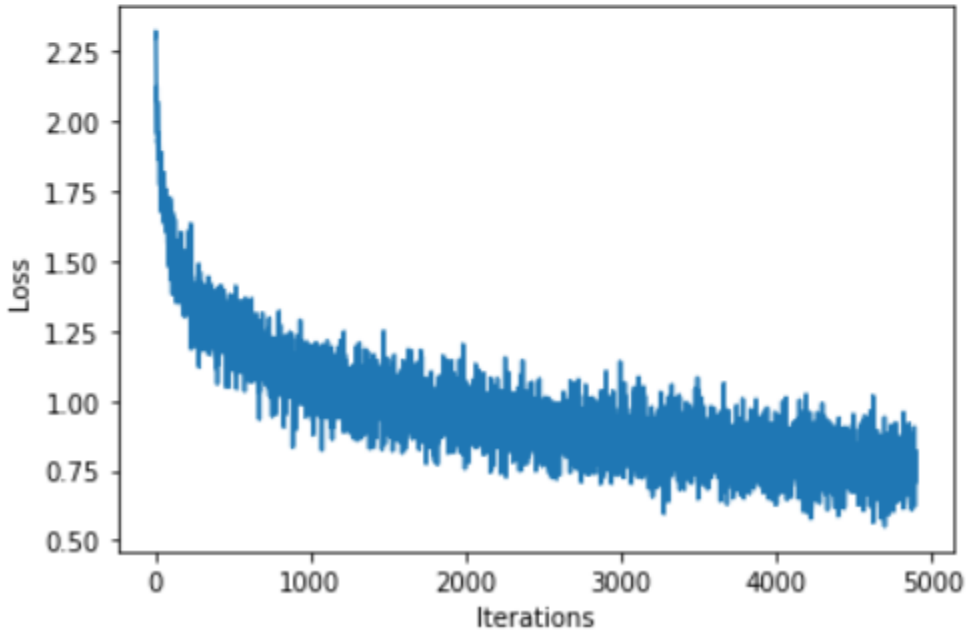
P = 0.5


Validation Accuracy
80
75
70
65
60
55
50
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs

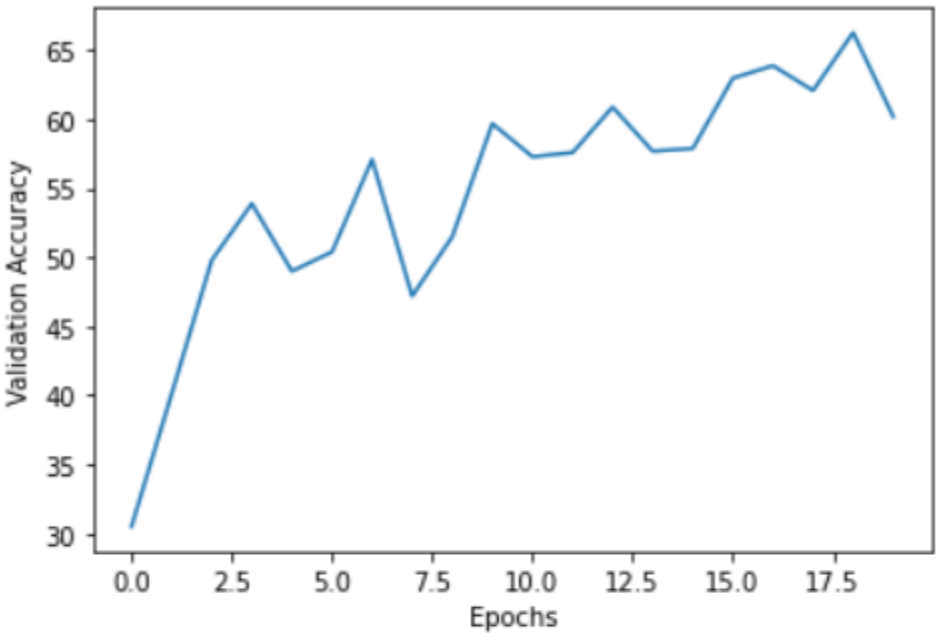

Loss
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0
1000
2000
3000
4000
5000
Iterations

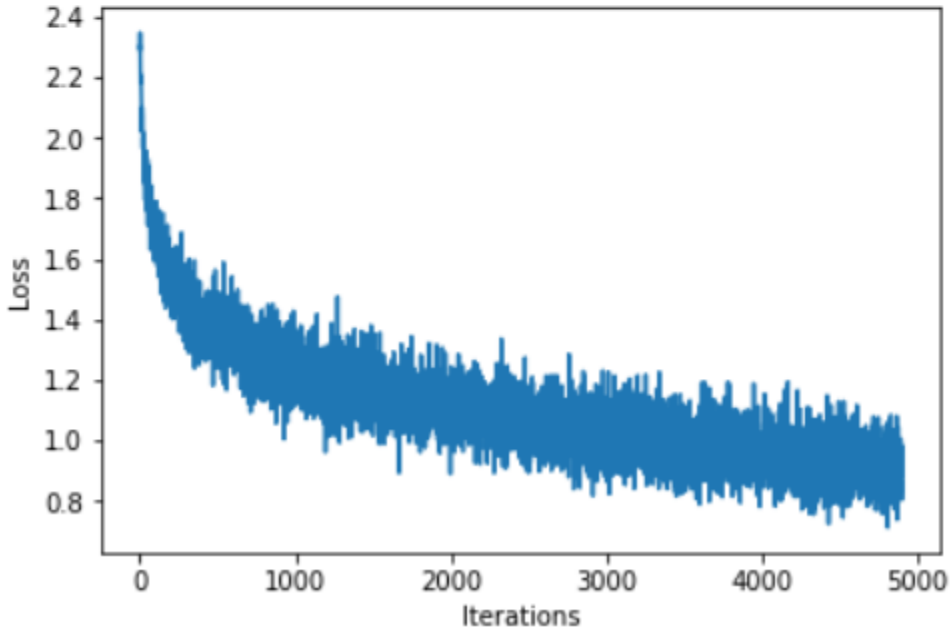
p = 0.6


Validation Accuracy
70
60
50
40
30
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs

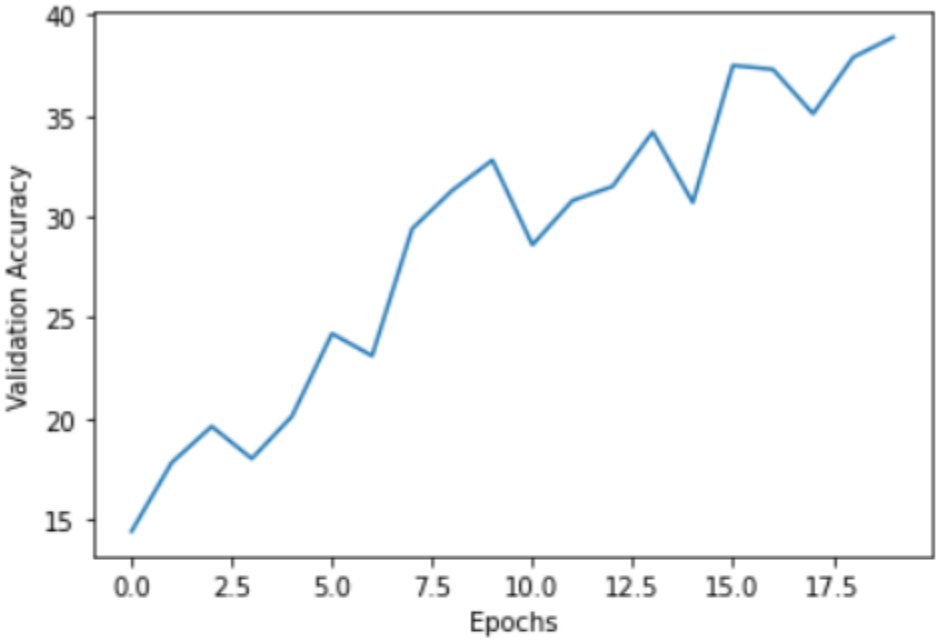

Loss
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0
1000
2000
3000
4000
5000
Iterations

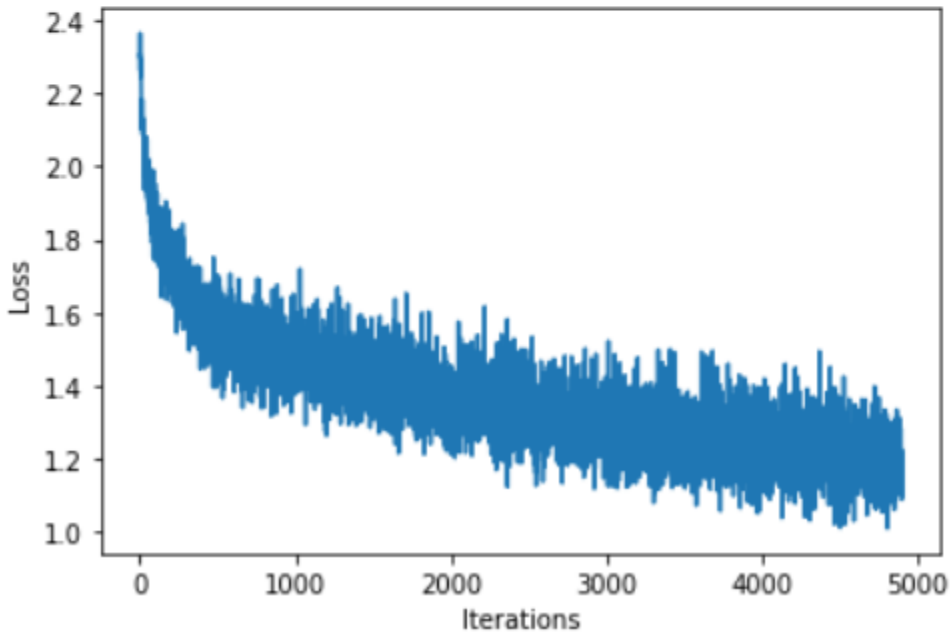
P = 0.7


Validation Accuracy
65
60
55
50
45
40
35
30
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs

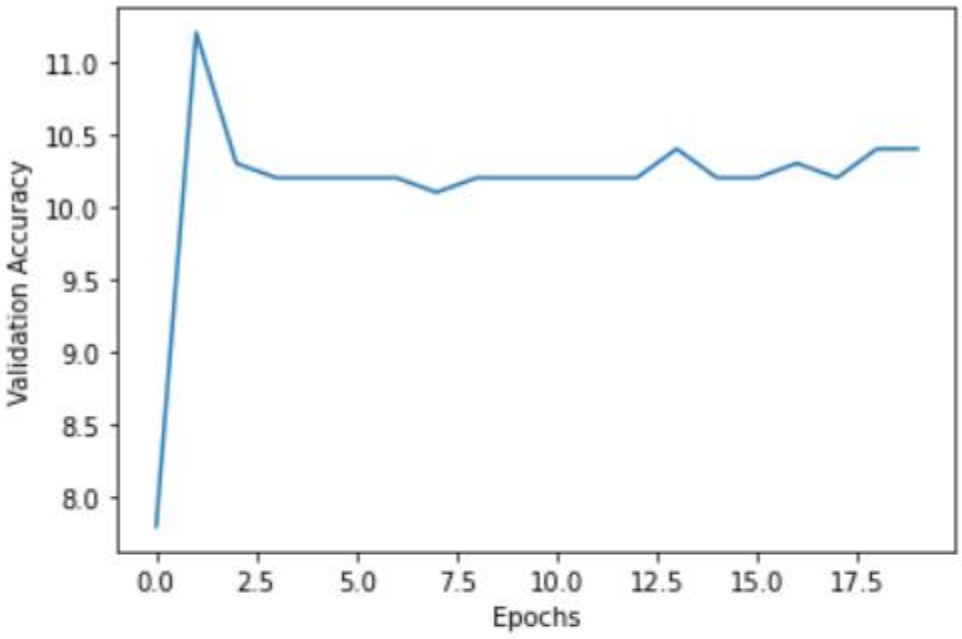

Loss
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0
1000
2000
3000
4000
5000
Iterations

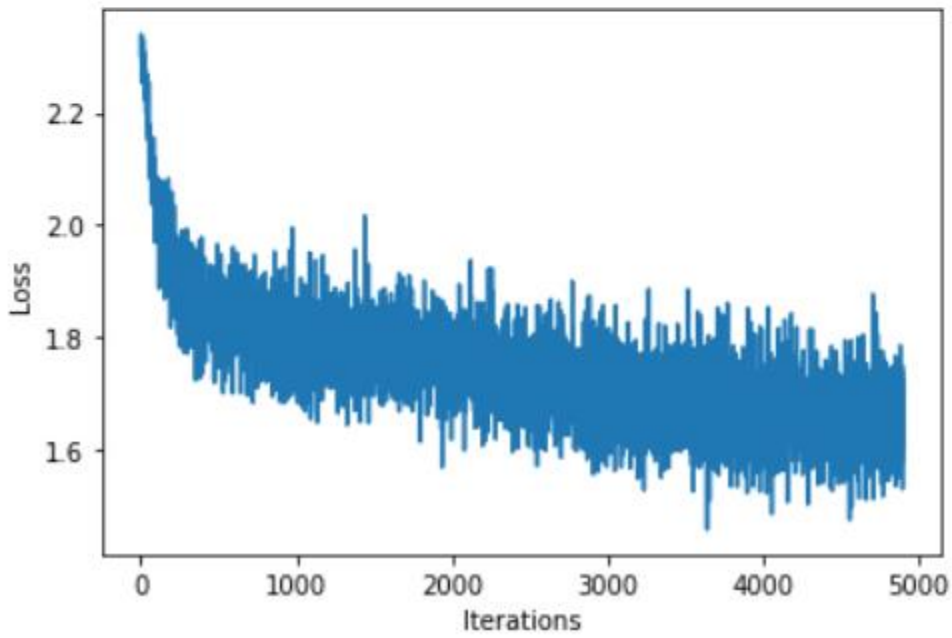
P = 0.8


Validation Accuracy
Epochs
40
35
30
25
20
15
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Loss
Iterations
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0
1000
2000
3000
4000
5000

P = 0.9


Validation Accuracy
11.0
10.5
10.0
9.5
9.0
8.5
8.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs


Loss
2.2
2.0
1.8
1.6
0
1000
2000
3000
4000
5000
Iterations