# RBE549: Homework 0 - Alohomora

Chinmay Sunil Kate
Robotics Engineering Department
Worcester Polytechnic Institute
cskate[at]wpi.edu

## I. PHASE I

In Phase 1. Boundary detection algorithm is implemented. A simple method to detect boundary is to look for intensity discontinuity in an image called edges. Probability of boundary lite (pb_lite) algorithm is been implemented along with the Canny and Sobel baselines. pb_lite considers texture and color discontinuity along with intensity discontinuity. It mostly suppress False positives that the classical method produces. Figure 1 descibes about the entire pipeline of the pb_lite architecture. This method contains following subsections:

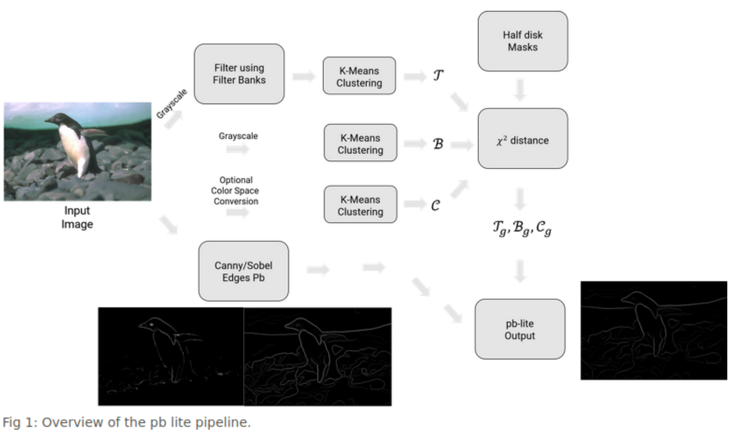


Fig. 1: Overview of pb_lite pipeline

### A. Filter Bank Generation

There are 3 types of filters that are used in this algorithm: Oriented derivative of Gaussian (DoG) filter, Leung-Malik filter, and Gabor filter. Illustration of these filters are presented in figure 1, 2, 3. Oriented DoG filter is convolved with Sobel filters and Sobel Kernel and the result is rotated. LM filters uses first and second order Gaussian derivatives, Gaussians and Laplacian of Gaussians. Gabor filter uses Gaussian kernel filter modulated by sinusoidal plane wave.

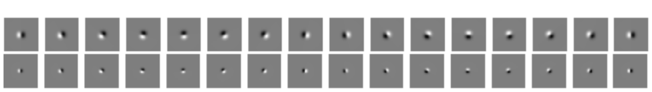

Fig. 2: Oriented derivative of Gaussian filter, generated with $\sigma$ = 1; 3; 5; 7.

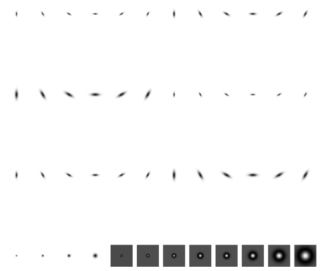

Fig. 3: Large Leung-Malik filter.

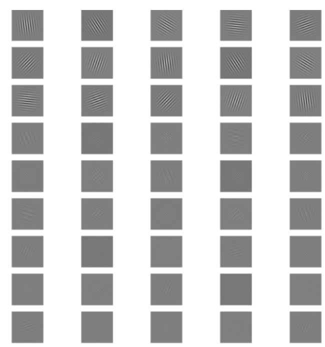

Fig. 4: Gabor filter.

### B. Texton, Brightness, Color Map Generation

Input image is filtered with filter banks. Responses of filtered image is clustered with KMeans function and is then vectorized to get Texton, Brightness and color map. Texton, brightness, and color map of all provided images are illustrated from figure 4 to figure 13.

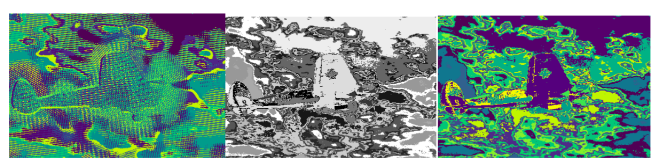

Fig. 5: $\mathcal{T}$, $\mathcal{B}$, $\mathcal{C}$ of image 2

### C. Texton, Brightness, Color Gradient Generation

Gradients for each map are computed using difference of values across different shape and sizes using Half-disc masks. In order to compute the texture, brightness, and color gradient map, we have to use the idea of half-disc masks. This map helps to compute Chi-square distance on gradient maps which

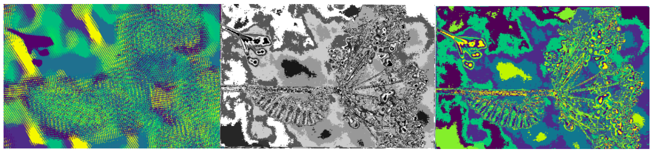

Fig. 6: $\mathcal{T}$, $\mathcal{B}$, $\mathcal{C}$ of image 3

Fig. 7: $\mathcal{T}$, $\mathcal{B}$, $\mathcal{C}$ of image 4

Fig. 8: $\mathcal{T}$, $\mathcal{B}$, $\mathcal{C}$ of image 5

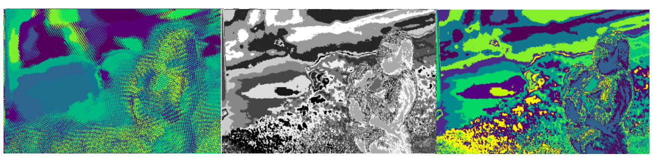

Fig. 9: $\mathcal{T}$, $\mathcal{B}$, $\mathcal{C}$ of image 6

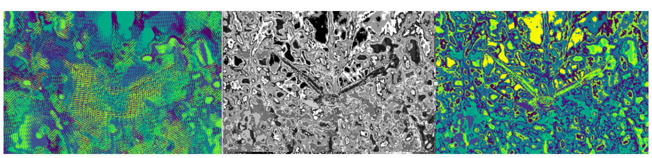

Fig. 10: $\mathcal{T}$, $\mathcal{B}$, $\mathcal{C}$ of image 7

Fig. 11: $\mathcal{T}$, $\mathcal{B}$, $\mathcal{C}$ of image 8

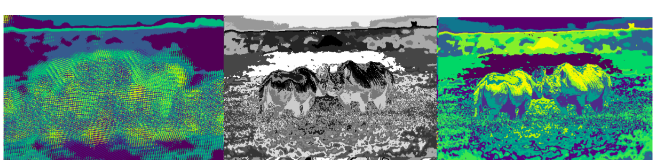

Fig. 12: $\mathcal{T}$, $\mathcal{B}$, $\mathcal{C}$ of image 9

is better solution than looping over each pixels. These masks are displayed in figure 14.

These gradient maps $\mathcal{T}_g$, $\mathcal{B}_g$ and $\mathcal{C}_g$ of all provided images are displayed from figure 15 to figure 24.

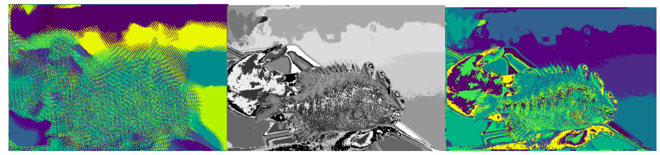

Fig. 13: $\mathcal{T}$, $\mathcal{B}$, $\mathcal{C}$ of image 10

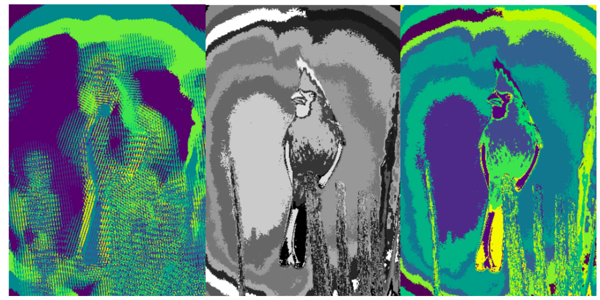

Fig. 14: $\mathcal{T}$, $\mathcal{B}$, $\mathcal{C}$ of image 10

Fig. 15: Half-disc masks generated with radius 1, 3, 5, 7.

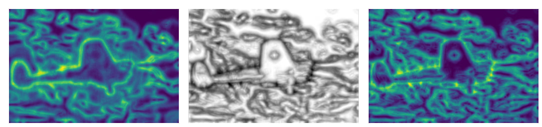

Fig. 16: $\mathcal{T}_g$, $\mathcal{B}_g$, $\mathcal{C}_g$ of image 1

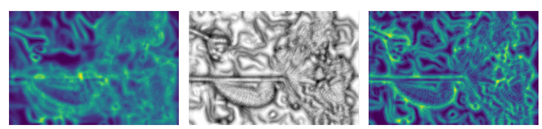

Fig. 17: $\mathcal{T}_g$, $\mathcal{B}_g$, $\mathcal{C}_g$ of image 2

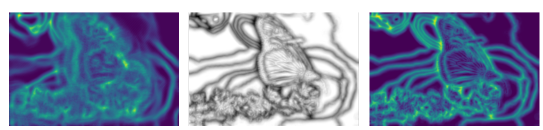

Fig. 18: $\mathcal{T}_g$, $\mathcal{B}_g$, $\mathcal{C}_g$ of image 3

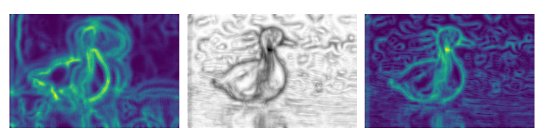

Fig. 19: $\mathcal{T}_g$, $\mathcal{B}_g$, $\mathcal{C}_g$ of image 4

### *D. Boundary Detection*

The Mean of gradient maps are calculated and aggregated dynamically with the Sobel and Canny baseline methods. The results of all 10 provided images are illustrated from figure 25 to 34.

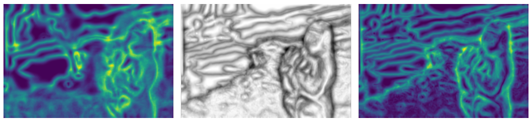

Fig. 20: $\mathcal{T}_g$, $\mathcal{B}_g$, $\mathcal{C}_g$ of image 5

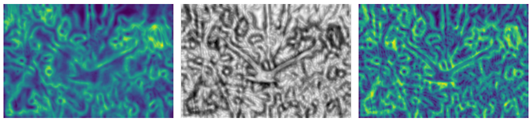

Fig. 21: $\mathcal{T}_g$, $\mathcal{B}_g$, $\mathcal{C}_g$ of image 6

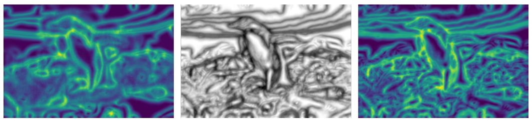

Fig. 22: $\mathcal{T}_g$, $\mathcal{B}_g$, $\mathcal{C}_g$ of image 7

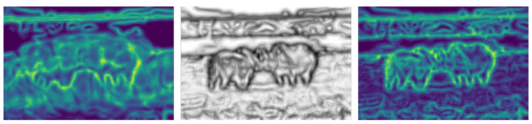

Fig. 23: $\mathcal{T}_g$, $\mathcal{B}_g$, $\mathcal{C}_g$ of image 8

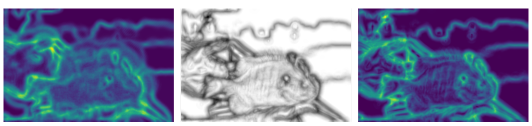

Fig. 24: $\mathcal{T}_g$, $\mathcal{B}_g$, $\mathcal{C}_g$ of image 9

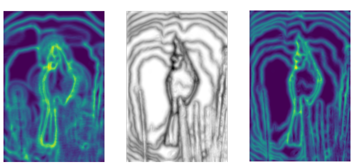

Fig. 25: $\mathcal{T}_g$, $\mathcal{B}_g$, $\mathcal{C}_g$ of image 10

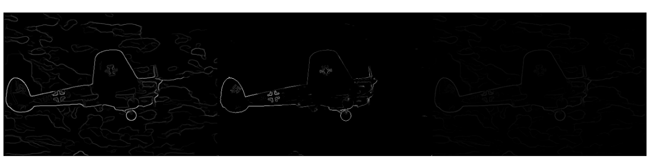

Fig. 26: Sobel, Canny, and pb-lite result of image 1

Fig. 27: Sobel, Canny, and pb-lite result of image 2

Fig. 28: Sobel, Canny, and pb-lite result of image 3

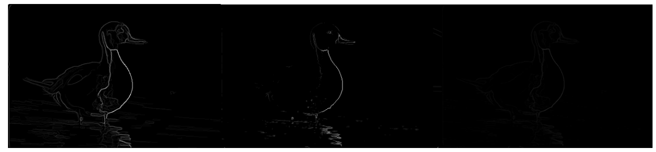

Fig. 29: Sobel, Canny, and pb-lite result of image 4

Fig. 30: Sobel, Canny, and pb-lite result of image 5

Fig. 31: Sobel, Canny, and pb-lite result of image 6

Fig. 32: Sobel, Canny, and pb-lite result of image 7

Fig. 33: Sobel, Canny, and pb-lite result of image 8

Fig. 34: Sobel, Canny, and pb-lite result of image 9

### *E. Result Analysis*

From the output we can seen that Canny method has many False positive and the detection is very sharp. In Sobel method we see the boundary detection is too suppressed. We take dynamic weights of the world(Canny and Sobel methods) and aggregate with the results of pb_lite outputs. This gives really impressive reasults as we see in the figure.

Fig. 35: Sobel, Canny, and pb-lite result of image 10

# II. PHASE II

## A. Introduction

In Phase 2. Multiple Neural Architecture needs to be implemented on CIFAR10 dataset. CiFAR10 Dataset has 10 classes and has images of 50,000 with the size of 32X32. Needs to implement custom models, Resnet, ResNext, Densenet architectures. with various augmentation and standardization techniques.

## B. First Neural Network

CNN on CIFAR10 dataset with custom model has very basic architecture with 2 convolution layers, 2 pooling, RELU as activation function and 3 fully connected layers. Hyperparameters are as follows 50 Epochs, 100 Mini-Batch, learning_rate = 0.01. Got Accuracy 67% on Train Dataset and 68.5% on Test Dataset. Total Number of Parameters in Basic model is 62006.

```
[3453  107  188  101  110   45   81   89  555  271] (0)
[ 153 3615   34   59   20   31   80   49  280  679] (1)
[ 456   29 2521  365  462  266  528  208   88   77] (2)
[ 143   34  246 2533  279  725  634  210   99   97] (3)
[ 156   16  289  306 2911  148  462  574   67   71] (4)
[  61   21  222 1209  224 2507  302  345   46   63] (5)
[  45   32  185  280  166   91 4076   43   39   43] (6)
[  78   15  151  284  279  247  121 3683   31  111] (7)
[ 296  116   52   84   24   28   37   16 4149  198] (8)
[ 166  261   33   69   29   29   76   96  189 4052] (9)
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
```

Fig. 36: Confusion Matrix for First Neural Network Training Set (Accuracy: 67.21%)

```
[776  19  29  15  14  10   8  14  72  43] (0)
[ 28 767   7  10   4   7  14   2  34 127] (1)
[ 92   4 501  69  81  72 112  39  12  18] (2)
[ 35   8  43 483  67 168 104  51  19  22] (3)
[ 26   2  64  69 587  38  82 108  17   7] (4)
[ 21   0  39 216  33 562  54  57   6  12] (5)
[ 14   4  31  51  24  27 829  12   5   3] (6)
[ 16   3  39  55  41  75  21 724   4  22] (7)
[ 82  24   8  16   5   9  10   3 798  45] (8)
[ 28  52   9  17   2   5   9  18  37 823] (9)
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
Accuracy: 68.5 %
```

Fig. 37: Confusion Matrix for First Neural Network Testing Set (Accuracy: 68.5%)

## C. Improved Neural Network

Modified_Network has many changes into the architecture as well as into the augmentation of Dataset. In Architecture BatchNorm Layers are been added and Network is more Deep. In Augmentation techniques Horizontalflip, Padding, RandomCrop, Normalization of data is done. Hyperparameters are as follows 50 Epochs, 100 Mini-Batch, learning_rate = 0.01. Got Accuracy 92% on Train Dataset and 83.52% on Test Dataset. There was some overfitting with this architecture at the given epochs as We get the accuracy but loss didn't reduced much. Total Number of Parameters in Modified model is 5852234.

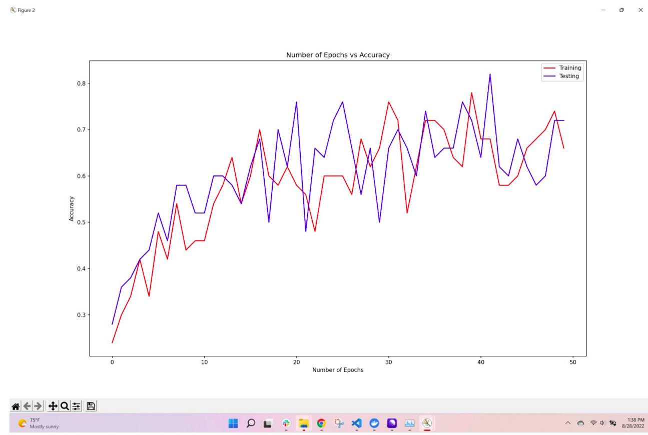


Fig. 38: Accuracy vs No. of Epochs for First Neural Network on Training/Testing Set

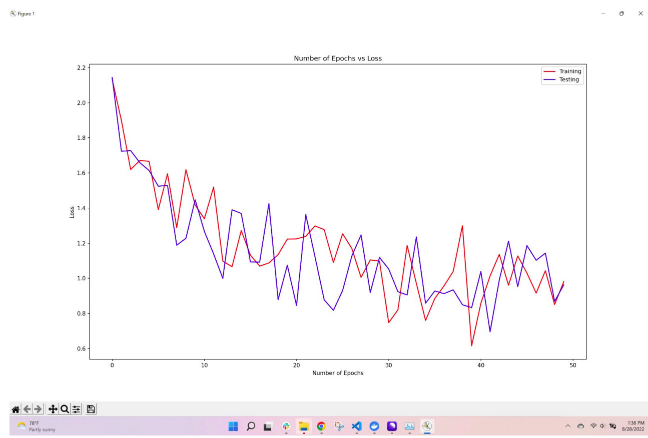


Fig. 39: Loss vs No. of Epochs for First Neural Network on Training/Testing Set

```
[4176   36  239   13   46    8   25    6  307  144] (0)
[  16 4678   11    4    4    5   12    2  110  158] (1)
[ 113    6 4525   33  116   22  131    7   30   17] (2)
[  63   10  450 3324  270  371  336   62   71   43] (3)
[  24    0  186   38 4574   22  109   19   20    8] (4)
[  23    2  313  439  230 3730  152   74   18   19] (5)
[  15    4  168   28   49    8 4697    3   20    8] (6)
[  27    3  207   57  403  106   26 4120   24   27] (7)
[  54   17   31    3   14    3    4    0 4847   27] (8)
[  18   77   15    7   10    2   11    4   81 4775] (9)
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
```

Fig. 40: Confusion Matrix for Improved Neural Network Training Set (Accuracy: 92.452%)

```
[801  10  65   5  10   1   4   1  71  32] (0)
[  2 927   1   1   1   0   2   0  21  45] (1)
[ 29   0 863   9  39   9  31   5   7   8] (2)
[ 19   5 126 612  56  61  67  14  22  18] (3)
[  5   1  57  11 872   6  36   6   5   1] (4)
[  8   2 103 105  60 672  22  15   5   8] (5)
[  6   2  56   8  16   3 899   3   7   0] (6)
[ 10   2  72  16  81  29   2 771   6  11] (7)
[ 27   3   7   0   1   0   3   0 948  11] (8)
[  5  27   6   1   1   0   1   0  33 926] (9)
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
```

Fig. 41: Confusion Matrix for Improved Neural Network Testing Set (Accuracy: 83.52%)

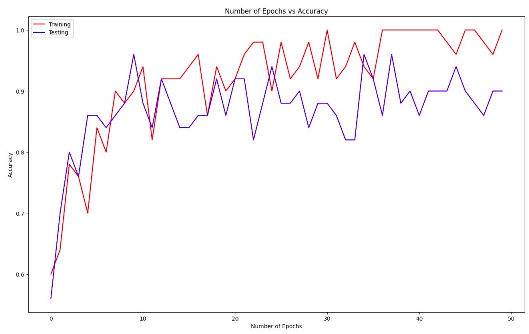


Fig. 42: Accuracy vs No. of Epochs for Improved Neural Network on Training/Testing Set

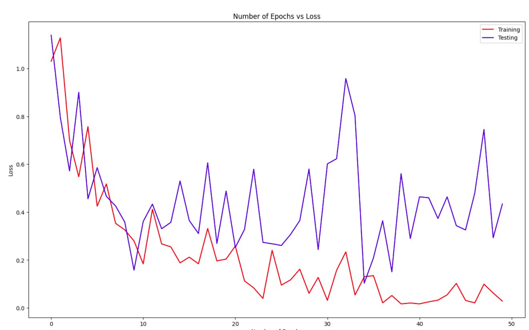


Fig. 43: Loss vs No. of Epochs for Improved Neural Network on Training/Testing Set

*D. ResNet*

ResNet is a famous NN architecture which uses Residual Layers as skip connections, This network has definitely improved accuracy with respect to loss. It uses 3 layers of 16,32,64 Residual blocks. In Augmentation techniques Horizontalflip, Padding, RandomCrop, Normalization of data is done. Hyperparameters are as follows 50 Epochs, 100 Mini-Batch, learning_rate = 0.01. Got Accuracy 86.8% on Train Dataset and 82.91% on Test Dataset. Total number of Parameters in Resnet Model is 1957384.

```
[4176   36  239   13   46    8   25    6  307  144] (0)
[  16 4678   11    4    4    5   12    2  110  158] (1)
[ 113    6 4525   33  116   22  131    7   30   17] (2)
[  63   10  450 3324  270  371  336   62   71   43] (3)
[  24    0  186   38 4574   22  109   19   20    8] (4)
[  23    2  313  439  230 3730  152   74   18   19] (5)
[  15    4  168   28   49    8 4697    3   20    8] (6)
[  27    3  207   57  403  106   26 4120   24   27] (7)
[  54   17   31    3   14    3    4    0 4847   27] (8)
[  18   77   15    7   10    2   11    4   81 4775] (9)
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
```

Fig. 44: Confusion Matrix for ResNet Training Set (Accuracy: 86.8%)

```
[801  10  65   5  10   1   4   1  71  32] (0)
[  2 927   1   1   1   0   2   0  21  45] (1)
[ 29   0 863   9  39   9  31   5   7   8] (2)
[ 19   5 126 612  56  61  67  14  22  18] (3)
[  5   1  57  11 872   6  36   6   5   1] (4)
[  8   2 103 105  60 672  22  15   5   8] (5)
[  6   2  56   8  16   3 899   3   7   0] (6)
[ 10   2  72  16  81  29   2 771   6  11] (7)
[ 27   3   7   0   1   0   3   0 948  11] (8)
[  5  27   6   1   1   0   1   0  33 926] (9)
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
Accuracy: 82.91 %
```

Fig. 45: Confusion Matrix for ResNet Testing Set (Accuracy: 82.91%)

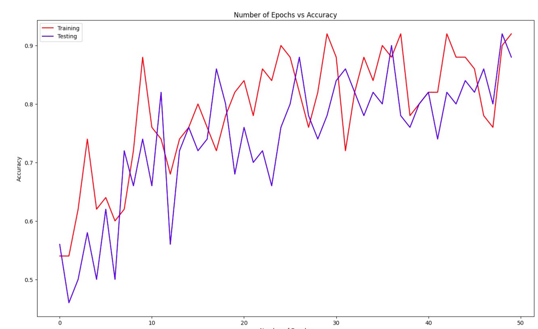


Fig. 46: Accuracy vs No. of Epochs for ResNet on Training/Testing Set

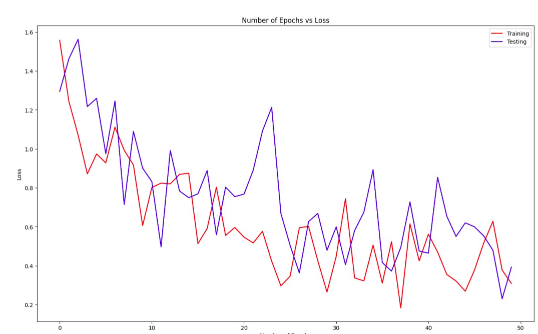


Fig. 47: Loss vs No. of Epochs for ResNet on Training/Testing Set

## E. ResNeXt

ResNext architecture on CIFAR10 dataset has ResNext layers where connections are splitted among the blocks and skip connection is used. This architecture has 4 ResNext layers with ResNext blocks and cardinality =2 and Bottleneck width of 64. Used weight_decay on SGD optimizer. In Augmentation techniques Horizontalflip, Padding, RandomCrop, Normalization of data is done. Hyperparameters are as follows 50 Epochs, 100 Mini-Batch, learning_rate = 0.01. Got Accuracy 95.8% on Train Dataset and 87.91% on Test Dataset.Total Number of parameters in ResNext is 9128778.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| [4929 | 0 | 20 | 10 | 15 | 0 | 1 | 5 | 19 | 1] | (0) |
| [ 36 | 4860 | 0 | 7 | 1 | 5 | 6 | 6 | 36 | 43] | (1) |
| [ 62 | 0 | 4652 | 64 | 65 | 20 | 76 | 53 | 8 | 0] | (2) |
| [ 18 | 0 | 11 | 4740 | 65 | 55 | 64 | 38 | 5 | 4] | (3) |
| [ 3 | 0 | 6 | 20 | 4926 | 5 | 10 | 30 | 0 | 0] | (4) |
| [ 3 | 0 | 14 | 386 | 82 | 4386 | 29 | 100 | 0 | 0] | (5) |
| [ 8 | 0 | 2 | 15 | 20 | 1 | 4942 | 6 | 5 | 1] | (6) |
| [ 7 | 0 | 3 | 28 | 29 | 4 | 1 | 4928 | 0 | 0] | (7) |
| [ 37 | 1 | 2 | 7 | 7 | 3 | 1 | 4 | 4937 | 1] | (8) |
| [ 51 | 17 | 5 | 20 | 2 | 1 | 11 | 23 | 42 | 4828] | (9) |
| (0) | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | |

Fig. 48: Confusion Matrix for ResNeXt Training Set (Accuracy: 95.8%)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| [930 | 1 | 14 | 12 | 8 | 0 | 2 | 2 | 27 | 4] | (0) |
| [ 19 | 907 | 0 | 2 | 0 | 0 | 5 | 3 | 23 | 41] | (1) |
| [ 28 | 0 | 794 | 38 | 45 | 20 | 45 | 22 | 7 | 1] | (2) |
| [ 13 | 1 | 19 | 820 | 40 | 49 | 27 | 20 | 9 | 2] | (3) |
| [ 5 | 0 | 10 | 17 | 928 | 2 | 20 | 17 | 0 | 1] | (4) |
| [ 6 | 0 | 7 | 142 | 38 | 761 | 10 | 36 | 0 | 0] | (5) |
| [ 5 | 0 | 8 | 24 | 13 | 3 | 941 | 3 | 3 | 0] | (6) |
| [ 10 | 0 | 5 | 19 | 30 | 8 | 0 | 928 | 0 | 0] | (7) |
| [ 21 | 2 | 2 | 7 | 2 | 0 | 3 | 1 | 957 | 5] | (8) |
| [ 24 | 15 | 4 | 7 | 1 | 0 | 5 | 5 | 19 | 920] | (9) |
| (0) | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | |

Accuracy: 88.86 %

Fig. 49: Confusion Matrix for ResNeXt Testing Set (Accuracy: 87.91%)

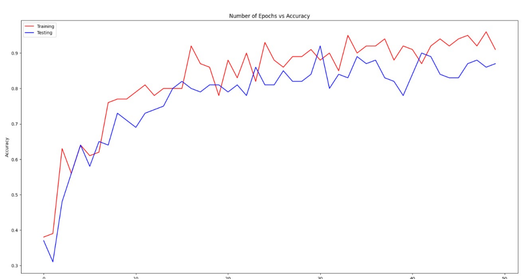


Fig. 50: Accuracy vs No. of Epochs for ResNeXt on Training/Testing Set

## F. DenseNet

DenseNet is really Popular architecture used to classify images on CIFAR10 dataset. It has 4 Dense layers with Transition of 6,12,24,16 and Bottleneck layers. It has Growth rate equals to 32. Adam optimizer being used with weight decay of 0.005. In Augmentation techniques Horizontalflip, Padding, RandomCrop, Normalization of data is done. Hyperparameters are as follows 50 Epochs, 100 Mini-Batch, learning_rate = 0.01. Got Accuracy 95.92% on Train Dataset and 88.51% on Test Dataset. Total Number of Parameters in DenseNet model is 6956298.

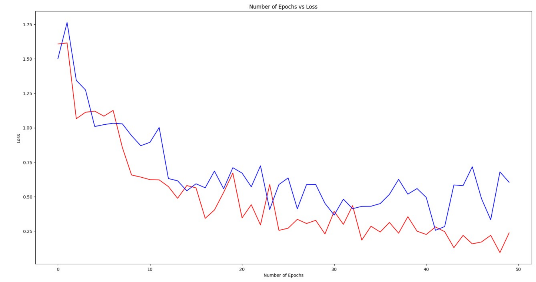


Fig. 51: Loss vs No. of Epochs for ResNeXt on Training/Testing Set

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| [4949 | 8 | 1 | 12 | 8 | 1 | 5 | 0 | 8 | 8] | (0) |
| [ 6 | 4970 | 1 | 4 | 1 | 0 | 1 | 0 | 3 | 14] | (1) |
| [ 135 | 6 | 4471 | 112 | 149 | 5 | 106 | 6 | 8 | 2] | (2) |
| [ 8 | 4 | 9 | 4860 | 52 | 26 | 34 | 4 | 0 | 3] | (3) |
| [ 7 | 0 | 3 | 18 | 4960 | 1 | 8 | 3 | 0 | 0] | (4) |
| [ 12 | 9 | 9 | 502 | 116 | 4252 | 41 | 50 | 2 | 7] | (5) |
| [ 3 | 4 | 3 | 37 | 24 | 1 | 4928 | 0 | 0 | 0] | (6) |
| [ 25 | 5 | 12 | 60 | 150 | 5 | 12 | 4715 | 4 | 12] | (7) |
| [ 34 | 7 | 0 | 4 | 1 | 0 | 5 | 0 | 4939 | 10] | (8) |
| [ 26 | 41 | 0 | 8 | 0 | 0 | 2 | 2 | 5 | 4916] | (9) |
| (0) | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | |

Fig. 52: Confusion Matrix for DenseNet Training Set (Accuracy: 95.92%)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| [930 | 6 | 7 | 11 | 11 | 0 | 5 | 1 | 20 | 9] | (0) |
| [ 6 | 973 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 17] | (1) |
| [ 49 | 3 | 736 | 62 | 79 | 7 | 51 | 4 | 5 | 4] | (2) |
| [ 14 | 5 | 11 | 856 | 36 | 26 | 31 | 11 | 4 | 6] | (3) |
| [ 7 | 1 | 3 | 12 | 955 | 3 | 15 | 3 | 0 | 1] | (4) |
| [ 5 | 3 | 10 | 173 | 48 | 727 | 13 | 16 | 0 | 5] | (5) |
| [ 5 | 2 | 5 | 32 | 14 | 2 | 936 | 1 | 1 | 2] | (6) |
| [ 19 | 2 | 7 | 35 | 54 | 5 | 4 | 865 | 1 | 8] | (7) |
| [ 33 | 7 | 3 | 6 | 2 | 0 | 1 | 0 | 934 | 14] | (8) |
| [ 13 | 32 | 0 | 3 | 0 | 0 | 3 | 0 | 10 | 939] | (9) |
| (0) | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | |

Accuracy: 88.51 %

Fig. 53: Confusion Matrix for DenseNet Testing Set (Accuracy: 88.51%)

## G. Analysis

Here from the results DenseNet truly gives the better accuracy and loss on Test dataset due to Transition layers and growth rate of layers. Followed By ResNext wich had bottleneck and split connections and skip connections. Surprisingly Modified NN got better results but still there was some overfitting on the Train Dataset, so easily Resnet performed well with the residual layers of skip connections. Finally Vanilla did performed well inspite of limitations discussed in the subsection.

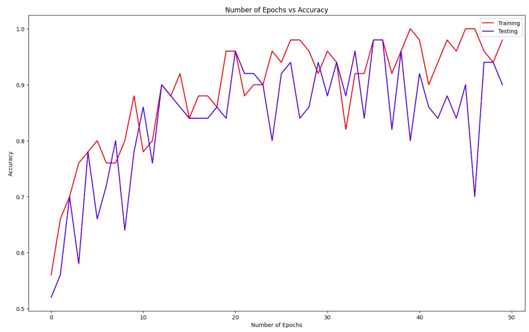


Fig. 54: Accuracy vs No. of Epochs for DenseNet on Training/Testing Set

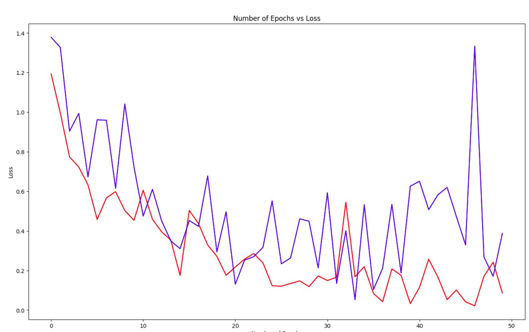


Fig. 55: Loss vs No. of Epochs for DenseNet on Training/Testing Set