HW0 : Alohomora

Thanikai Adhithiyan Shanmugam
Robotics Engineering Department
Worcester Polytechnic Insititute
(Using 1 Late Day)

***Abstract*—Boundary detection is an important, well-studied computer vision problem. It would be nice to have algorithms which know where one object transitions to another. This paper introduces a simplified version of the probability of boundary(pb) detection algorithm and compares with the classical Sobel and Canny edge detection algorithm baselines. This paper also trains and compares different neural network models with the CIFAR10 dataset with various criteria like number of parameters, and train and test set accuracies and provides a detailed analysis of why one architecture works better than another one.**

I. Phase 1

A. Filter Banks

The first step of the pb lite boundary detection pipeline is to filter the image with a set of filter banks. We will create three different sets of filter banks for this purpose: Difference of Gaussian(DoG), Leung-Maik Filters(LM), and Gabor Filters.

1) DoG: The DoG filter bank is created by convolving the Sobel Filter with the Gaussian Kernel and rotating it in different orientations. In this work, the filter bank contains 2 scales and 16 orientations constituting 32 filters.

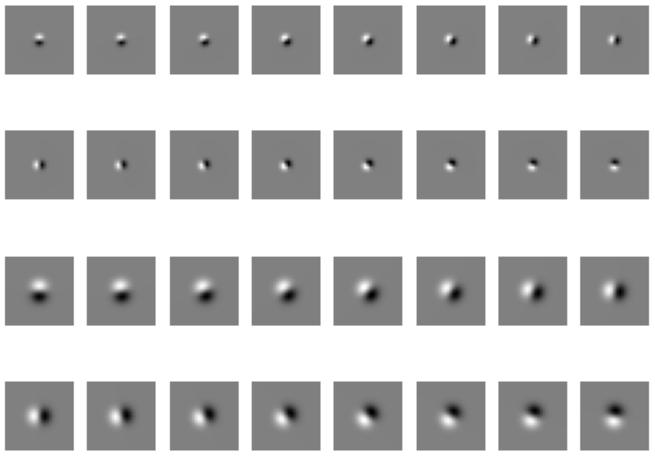


Fig. 1. DoG Filter Bank

2) Leung Malik: The Leung Malik(LM) filter bank consists of the 1st derivative and 2nd derivative Gaussian, Laplacian of Gaussian kernel and 2D Gaussian Kernel of different orientations and scales. In our work, we have 4 scales of $1, \sqrt{2}, 2, 2\sqrt{2}$ and 6 orientation for the 1st and 2nd derivatives. Since this work includes only Small LM filters, the Gaussians occur at the four basic scales while the 8 LOG filters occur at σ and 3σ which constitutes 60 filters in total.

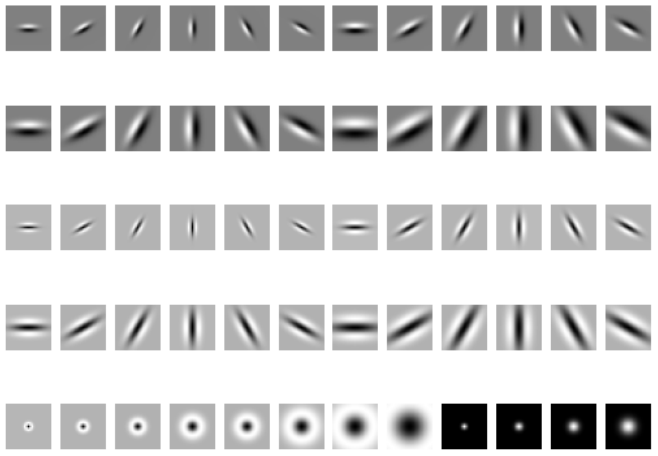


Fig. 2. LM Filter Bank

3) Gabor Filter: A typical gabor filter is a gaussian kernel function modulated by a sinusoidal plane wave. In this paper, the gabor filter has implemented along 5 scales of [2,3,4,5,6] and 8 orientations totalling 40 filters

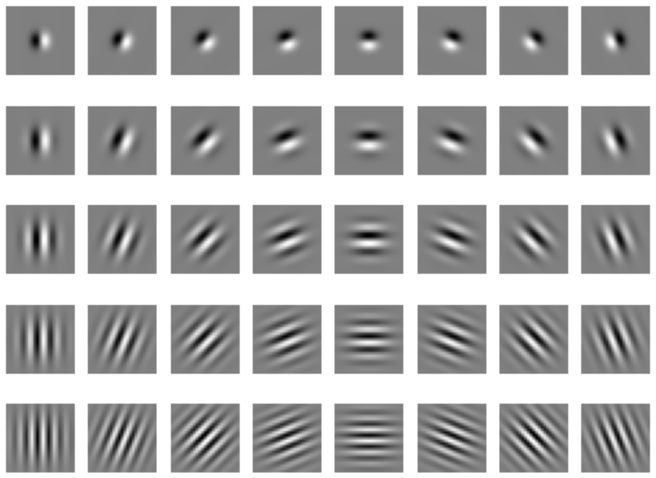


Fig. 3. Gabor Filter Bank

B. *Half Disc Masks*

Masking is an important approach in image processing and computer vision because it allows for the separation of certain objects or areas within an image. The paper implements Half Disc Masks. The half-disc masks are simply (pairs of) binary images of half-discs. Half disks masks in this paper are implemented by creating a gray scale semi-disk and rotating with different orientations. Here, we have used 4 radius or scales for generating the Mask.
Half disks are very important because it will allow us to

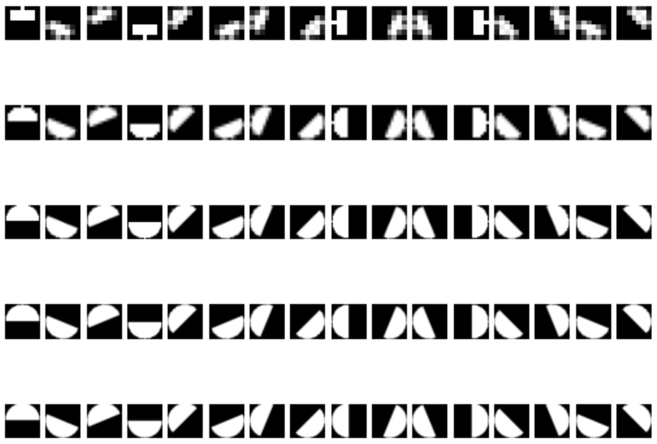


Fig. 4. Gabor Filter Bank

compute the 2(chi-square) distances (finally obtain values of Tg,Bg,Cg) using a filtering operation, which is much faster than looping over each pixel neighborhood and aggregating counts for histograms

C. *Textron, Color and Brightness Maps*

The Textron, colour and brightness as the name suggests, helps us gain information about the texture, color and brightness of the image at different areas respectively.

In this paper, the Textron map is created by passing the image through the combination of the filter bank discussed below and is normalised from (0,255). Since, there are 132 filters in total, we will have 132 dimensional vector corresponding to each pixel. The KMean clustering of 64 as suggested in the problem statement is implemented.

The concept of the brightness map is as simple as capturing the brightness changes in the image. THe brightness map is created by converting the image into a black and white image and the brightness values are clustered using KMeans. Here, the values are clustered in groups of 16.

The concept of the colour map is to capture the color changes or chrominance content in the image. THe brightness map is created by clustering the color values using KMeans. Here, the values are clustered in groups of 16.

D. *Textron, Color and Brightness Gradient Maps*

Tg, Bg, and Cg encode how much the texture, brightness and colour distributions change at a pixel. The values are

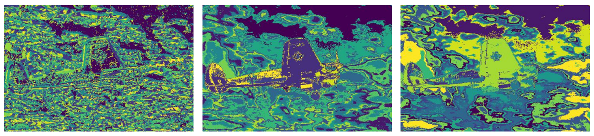


Fig. 5. Image1: Texton, Brightness, Color

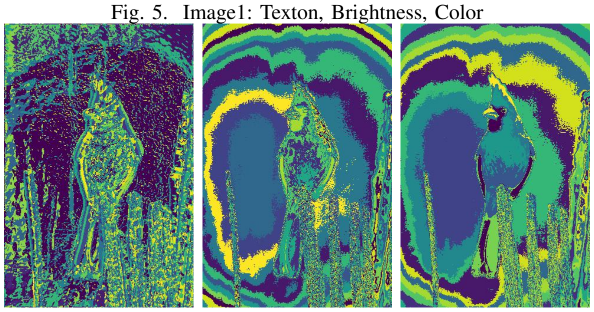


Fig. 6. Image2: Texton, Brightness, Color

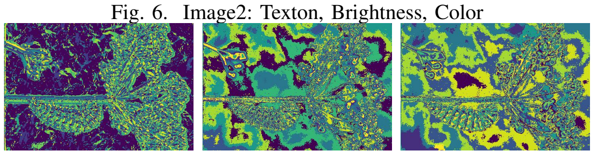


Fig. 7. Image3: Texton, Brightness, Color

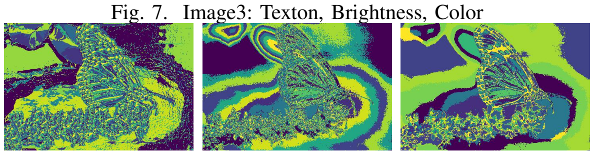


Fig. 8. Image4: Texton, Brightness, Color

calculated by comparing the values in the left and right pairs of half-disk masks.
The chi distance is a frequently used metric for comparing two histograms. Chi distance between two histograms g and h and the formula is used as mentioned in the problem statement.

$$\chi^2 = \frac{1}{2} sum_{n=1}^{K} \frac{(g-h)^2}{g+h+10^{-7}} \tag{1}$$

E. *Probability of boundary detection*

Using the calculated gradients and the Canny and Sobel baselines, the Pb is calculated as given in the problem statement as the results are listed below. The weights are taken as $[0.5, 0.5$ as instructed.

$$\chi^2 = \frac{T_g + B_g + C_g}{3}(w_1 Canny + w_2 Sobel) \tag{2}$$

II. Phase 2

In this section, we are exploring Neural Networks for classification on the CIFAR10 dataset. The dataset contains 50000 training images and 10000 training images which are labelled into 10 classes.
The Network used in this paper:
1) SimpleNet
2) Modified SimpleNet
3) ResNet

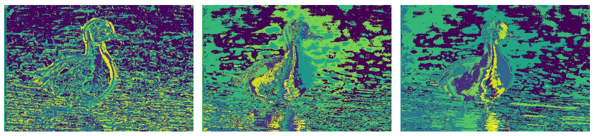


Fig. 9. Image5: Texton, Brightness, Color

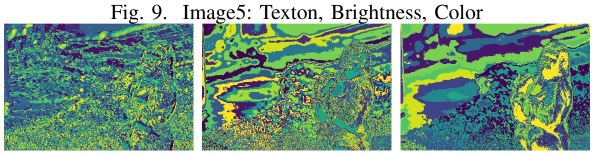


Fig. 10. Image6: Texton, Brightness, Color

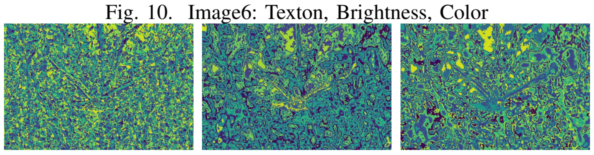


Fig. 11. Image7: Texton, Brightness, Color

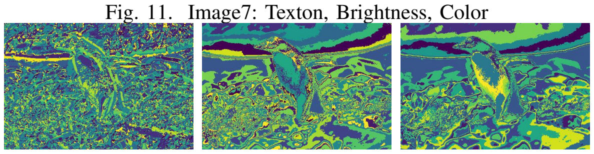


Fig. 12. Image8: Texton, Brightness, Color

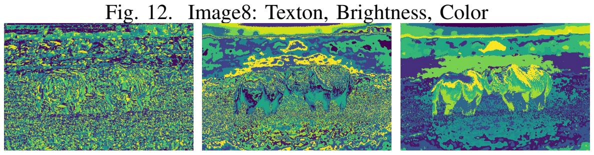


Fig. 13. Image9: Texton, Brightness, Color

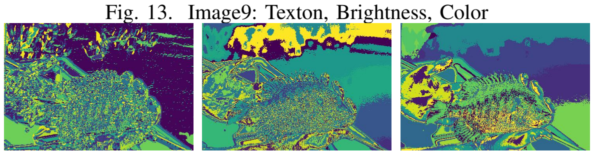


Fig. 14. Image10: Texton, Brightness, Color

4) ResNeXt
5) DenseNet
Let's discuss each model one by one

A. SimpleNet

The SimpleNet is a basic adaptation of CIFAR10 Model. The SimpleNet consists of 2 Convolutional Layers having a kernel size 3 and padding 1. Each layer uses ReLu activation function and max Padding is done which reduces the size of each output into half. Then, the model consists of a Flatten Layer which flattens the output from the convolution layer and sends it to 3 Fully Connected Layers of sizes 120,84,10 respectively. THe fully connected layers used the sigmoid activation function. The architecture of the Model is given below and the confusion matrix along with the accuracy of test and training are mentioned in the paper.

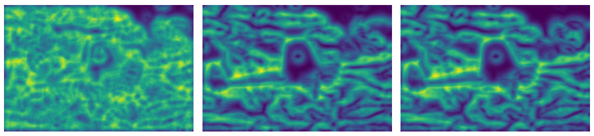


Fig. 15. Image1: Gradient Texton, Brightness, Color

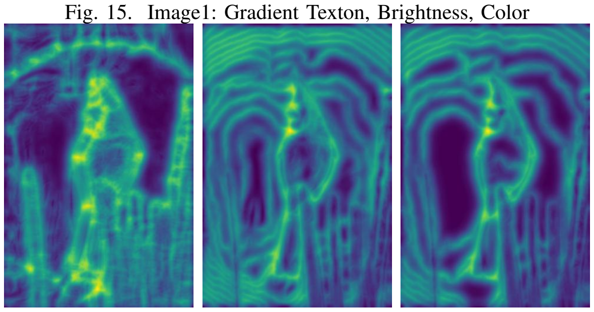


Fig. 16. Image2: Gradient Texton, Brightness, Color

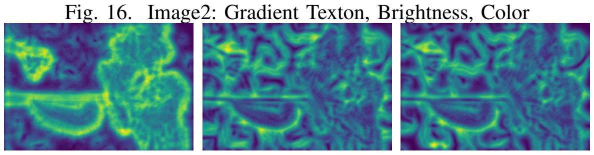


Fig. 17. Image3: Gradient Texton, Brightness, Color

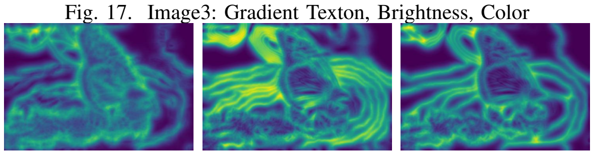


Fig. 18. Image4: Gradient Texton, Brightness, Color

B. Modified SimpleNet

The Modified SimpleNet is a basic adaptation of the CIFAR10 Model.The Modified has 2 extra layers compared to The SimpleNet is added with batch normalization to every layer including the convolutional and the fully connected layers. The hyperparameters remain the same whereas data augmentation has been added to the model and the model has been standardised. The image is rotated and cropped. The architecture of the Model is given below and the confusion matrix along with the accuracy of test and training are mentioned in the paper.

C. ResNet

The ResNet model implemented in this paper is an adaptation of the ResNet50 and ResNet100 architecture. This model starts with a convolutional layer of kernel size 3 and padding 1 and is then inputted into 2 Residual blocks which contain 2 convolutional layers with kernel size 3 and padding 1 with ReLu activation function. The dense blocks are then connected with 3 fully connected layers of size 120,84,10 with sigmoid and softmax activation functions respectively. The architecture of the Model is given below and the confusion matrix along with the accuracy of test and training are mentioned in the paper.

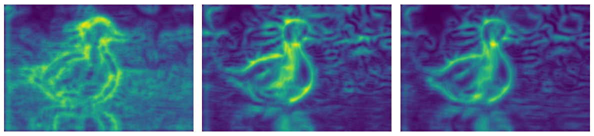


Fig. 19. Image5: Gradient Texton, Brightness, Color

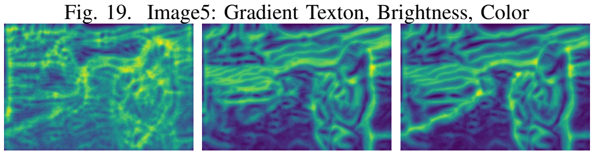


Fig. 20. Image6: Gradient Texton, Brightness, Color

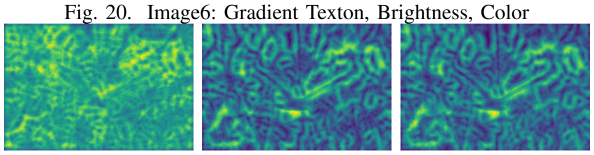


Fig. 21. Image7: Gradient Texton, Brightness, Color

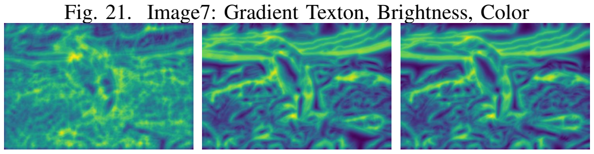


Fig. 22. Image8: Gradient Texton, Brightness, Color

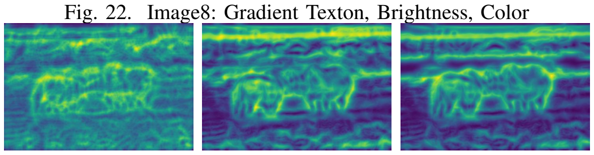


Fig. 23. Image9: Gradient Texton, Brightness, Color

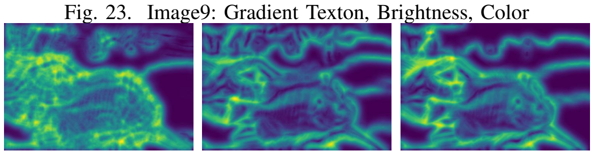


Fig. 24. Image10: Gradient Texton, Brightness, Color

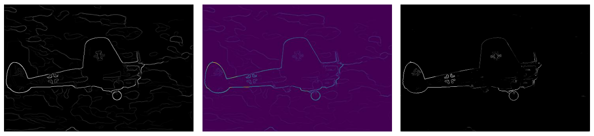


Fig. 25. Image1: Canny,Pb,Sobel

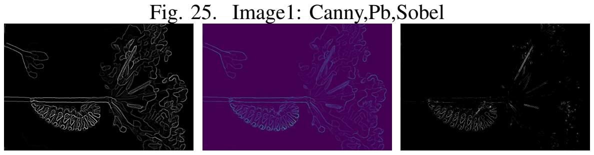


Fig. 26. Image2: Canny,Pb,Sobel

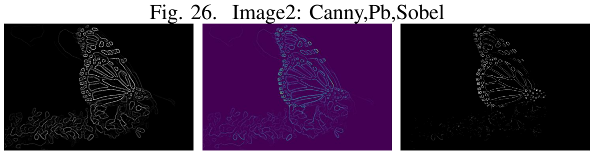


Fig. 27. Image3: Canny,Pb,Sobel

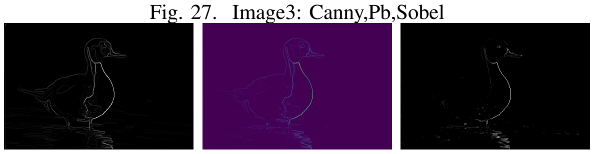


Fig. 28. Image4: Canny,Pb,Sobel

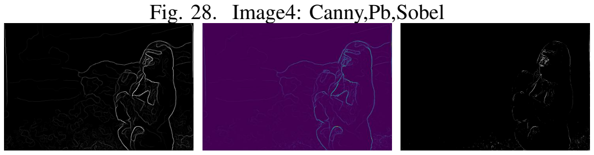


Fig. 29. Image5: Canny,Pb,Sobel

D. ResNeXt

The ResNet model implemented in this paper is an adaptation of the ResNeXt25 and ResNeXt59 architecture. This model starts with a convolutional layer of kernel size 3 and padding 1 and is then inputted into 2 ResNext blocks which contain 3 convolutional layers with kernel size 3 and padding 1 with ReLu activation function. The resnext blocks are connected by a single convolution layer having the same specs as the other convolution networks. The blocks are then connected with 3 fully connected layers of size 120,84,10 with sigmoid and softmax activation functions respectively. The architecture of the Model is given below and the confusion matrix along with the accuracy of test and training are mentioned in the paper.

E. DenseNet

The DenseNet model implemented in this paper is an adaptation of the DenseNet architecture. This model contains 2 dense blocks which contain 3 convolutional layers with kernel size 3 and padding 1 with ReLu activation function. The dense blocks are then connected with 3 fully connected layers of size 120,84,10 with sigmoid and softmax activation functions respectively. The architecture of the Model is given below and the confusion matrix along with the accuracy of test and training are mentioned in the paper.

F. Loss Function and Hyperparameters

The Loss function utilised for all the models in this paper is the cross entropy function. For optimizing the loss function, we use the Adam optimization which is a combination of RMSProp and momentum algorithms where the learning rate has been set to 0.001 and the beta value which constitutes square gradients (0.9,0.999). The dataset has been divided into min batches of 128 to improve faster training and efficiency. Each model has been trained over 20 epochs. The image is also augmented with cropping and flip techniques to increase data and efficiency for modified SimpleNet.

G. Results and Inferences

The train accuracy and the loss values was calculated over each during training of each models. DenseNet and ResNeXt learned the most as the train accuracies were 97.28 and 93.97 respectively. ResNet had more accuracy than ResNeXt but to exponentially larger number of parameters learned during training makes ResNeXt more efficient. SimpleNet as expected

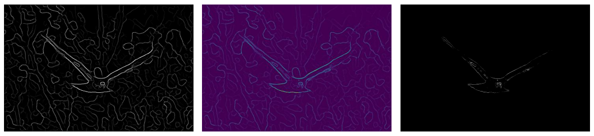


Fig. 30. Image6: Canny,Pb,Sobel

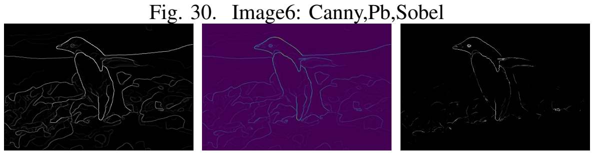


Fig. 31. Image7: Canny,Pb,Sobel

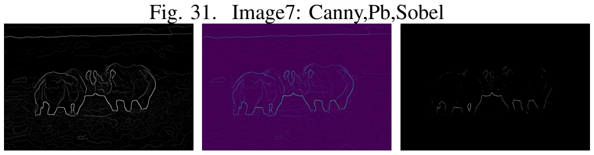


Fig. 32. Image8: Canny,Pb,Sobel

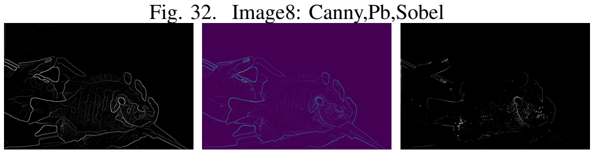


Fig. 33. Image9: Canny,Pb,Sobel

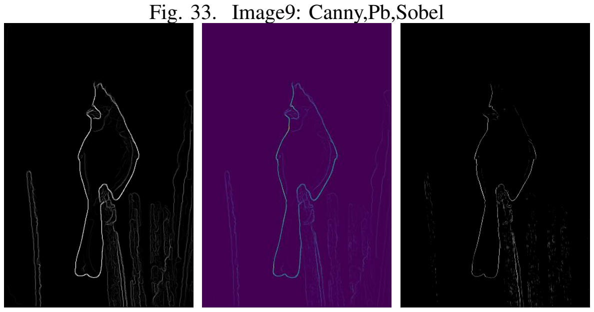


Fig. 34. Image10: Canny,Pb,Sobel

had an accuracy of 56.25 and the modified one had 69.53.// // The model was tested on 10,000 images of the CIFAR10 dataset. The accuracies of all the models were almost similar to each other. Modified SimpleNet did perform better than was anticipated which infers that batch normalization and augmentation is needed for this particular dataset to be classified better with Neural Network models. SimpleNet accuracy was the lowest among all as expected. The results of the models can be inferred from the table below.

III. Conclusion

A. Phase I

Comparison of Pb-lite detection algorithm with Canny and Sobel Baselines led to interesting results. We can infer that Pb-lite performs better than Sobel Baselines. Although, it is on par with Canny Baseline on performance, Pb-lite detects the edges better where Canny baseline has lot more false detection. Also, Pb-lite also the user to choose filter banks and upon selecting a more efficient strategy for selecting filters, Pb-lite surely outpeforms the other 2. //

B. Phase II

DenseNet outperforms the other models in terms of train accuracy. Although, the test accuracy is on par with ResNet and ResNeXt. Although SimpleNet outperforms others in terms of total parameters required for training, it still provides poor efficiency. ResNeXt is better than ResNet and DenseNet which has similar test efficiencies. It has shown that BatchNormalization is useful for this dataset as it increase the efficiency of image classification significantly. Also, augmentation of the data by cropping it and flipping it horizontally has proven to be useful. Efficiency can be still improved by increasing kernels in each convolution layer. Overall, on comparison among the model on CIFAR10 dataset, DenseNet proved to be the most efficient one followed by ResNeXt and ResNet. Although Modified SimpleNet gave good test accuracy, its training accuracy of the data is too low.

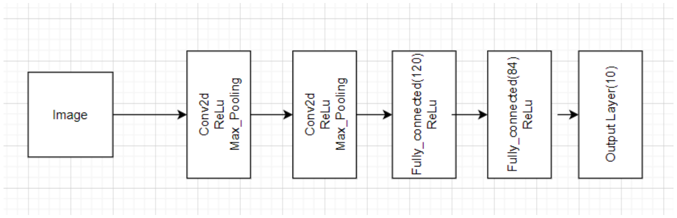


Fig. 35. SimpleNet Architecture

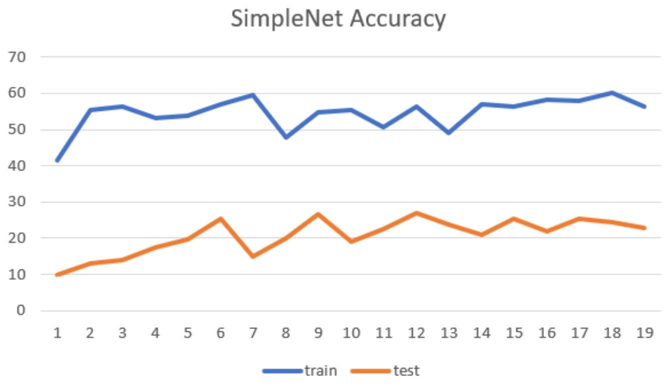


Fig. 36. SimpleNet Accuracies

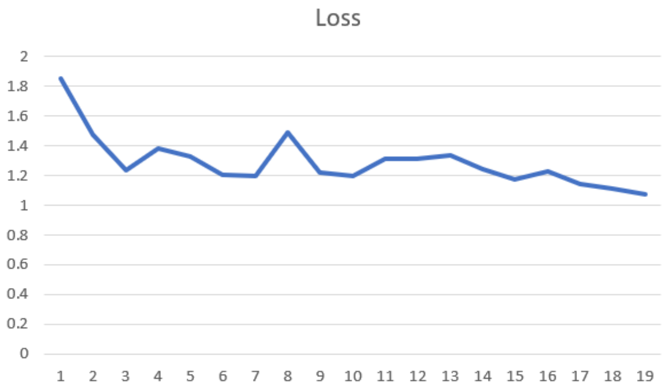


Fig. 37. SimpleNet Train Loss

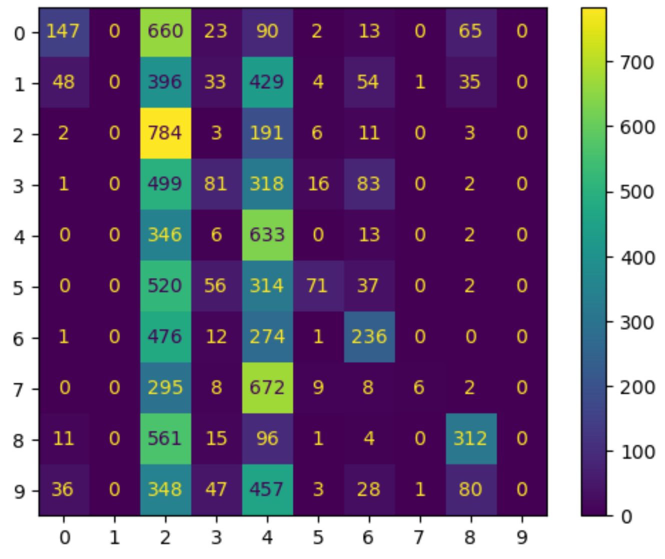


Fig. 38. SimpleNet Confusion Matrix

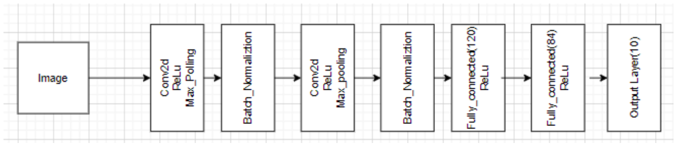


Fig. 39. Modified SimpleNet Architecture

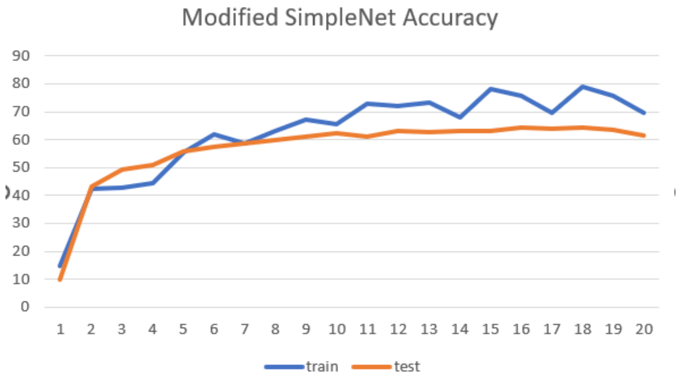


Fig. 40. Modified SimpleNet Accuracies

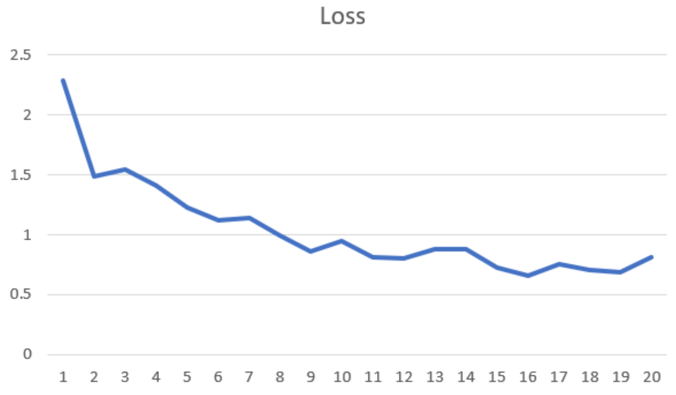


Fig. 41. Modified SimpleNet Train Loss

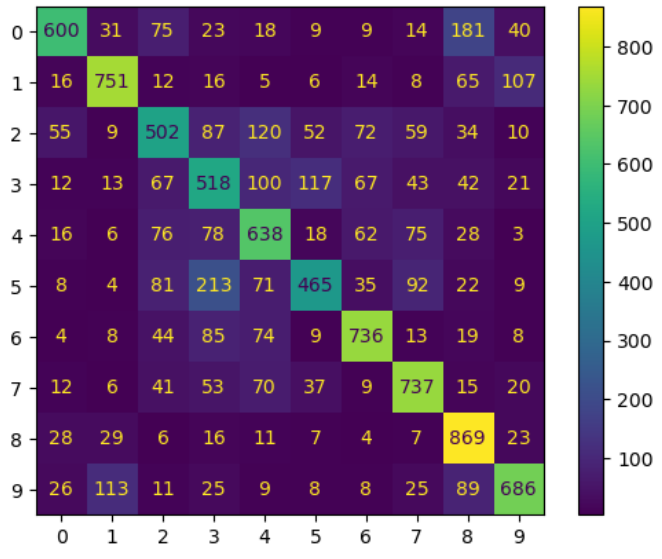


Fig. 42. Modified SimpleNet Confusion Matrix

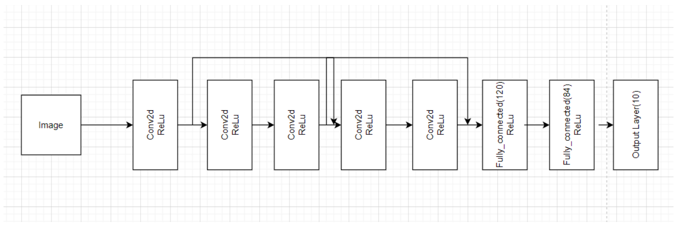


Fig. 43. ResNet Architecture

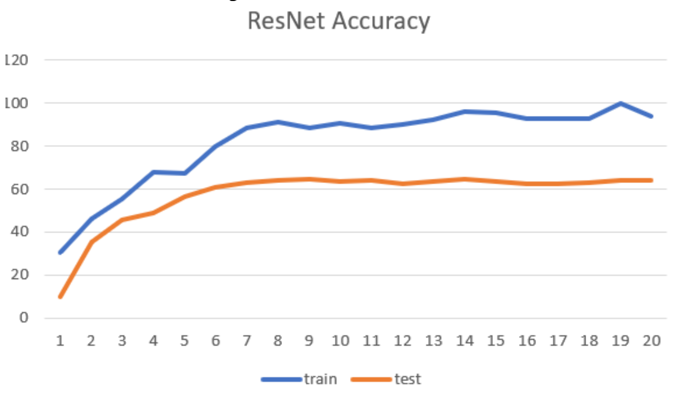


Fig. 44. ResNet Accuracies

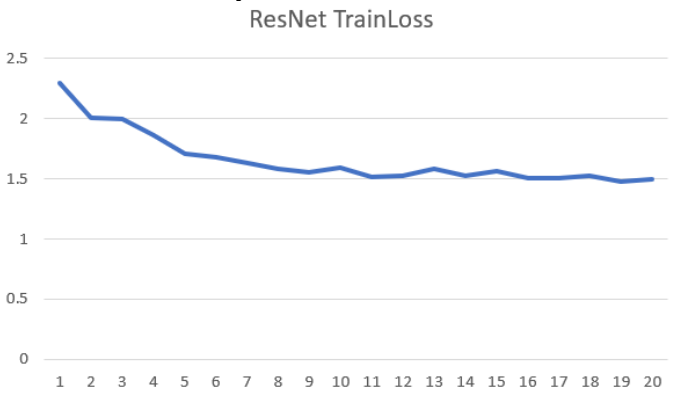


Fig. 45. ResNet Train Loss

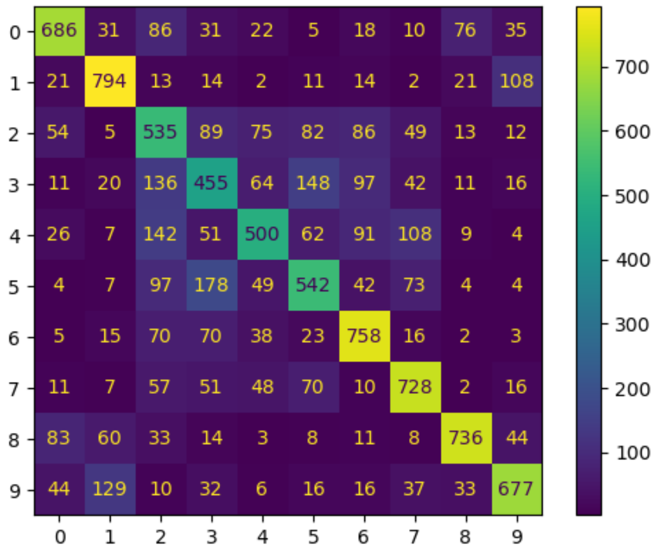


Fig. 46. ResNeXt Confusion Matrix

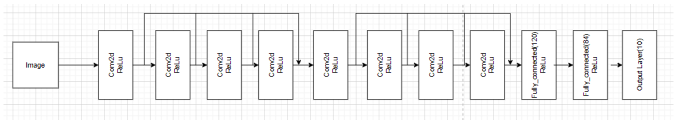


Fig. 47. ResNeXt Architecture

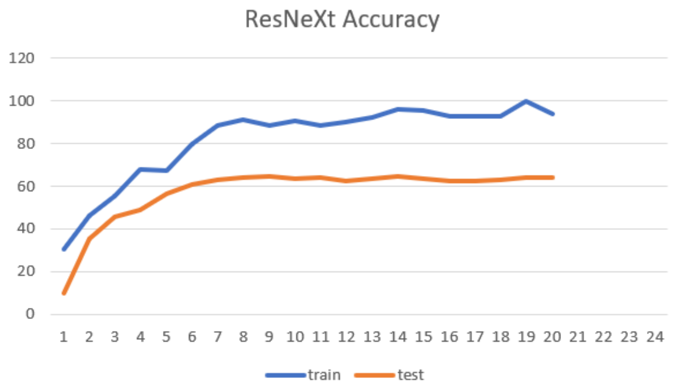


Fig. 48. ResNeXt Accuracies

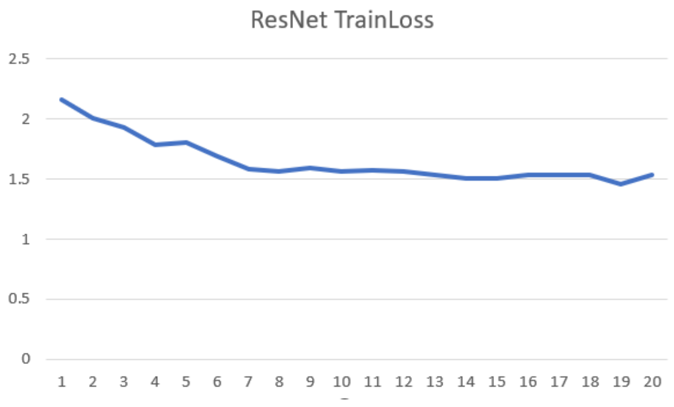


Fig. 49. ResNeXt Train Loss

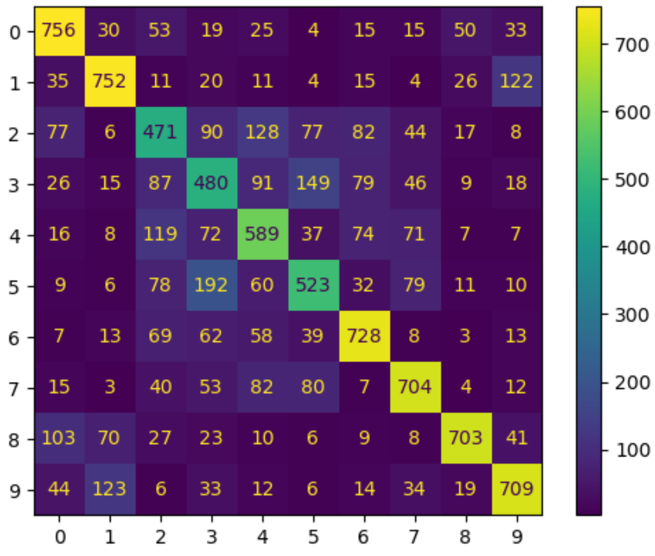


Fig. 50. ResNeXt Confusion Matrix

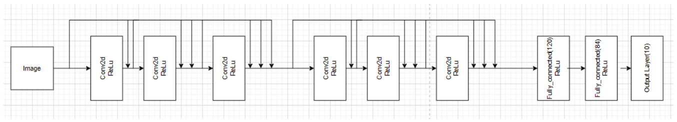


Fig. 51. DenseNet Architecture

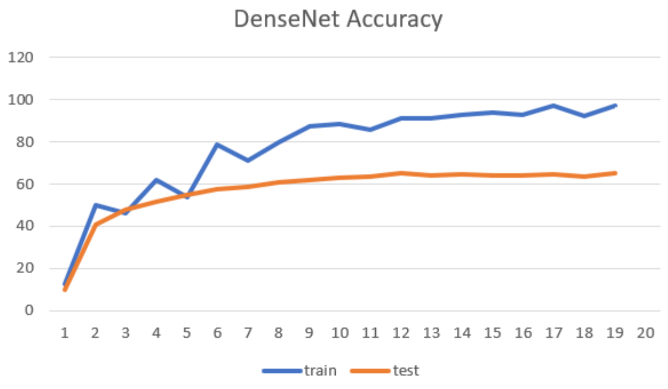


Fig. 52. DenseNet Accuracies

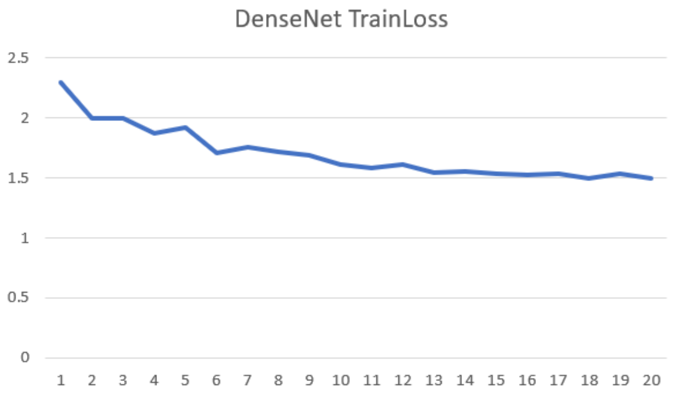
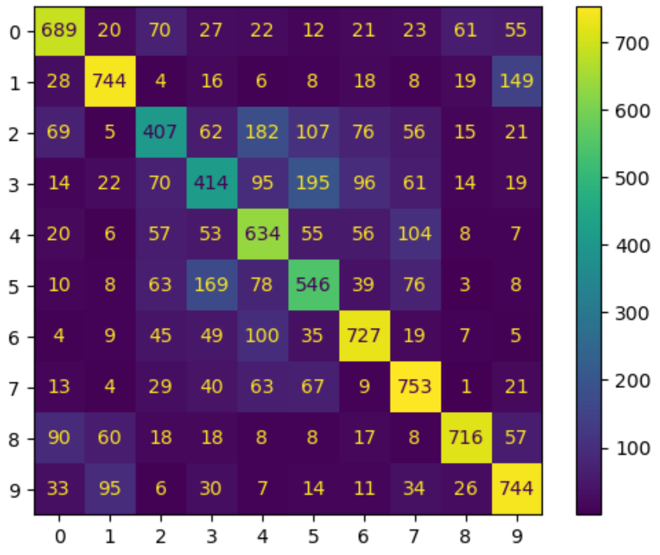
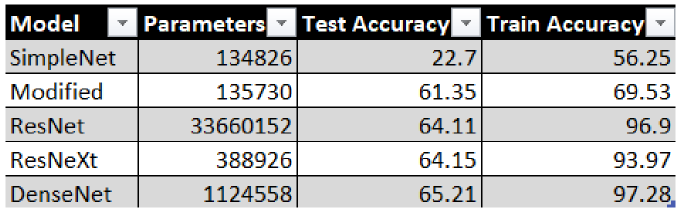


Fig. 53. DenseNet Loss Values



	0	1	2	3	4	5	6	7	8	9
0	689	20	70	27	22	12	21	23	61	55
1	28	744	4	16	6	8	18	8	19	149
2	69	5	407	62	182	107	76	56	15	21
3	14	22	70	414	95	195	96	61	14	19
4	20	6	57	53	634	55	56	104	8	7
5	10	8	63	169	78	546	39	76	3	8
6	4	9	45	49	100	35	727	19	7	5
7	13	4	29	40	63	67	9	753	1	21
8	90	60	18	18	8	8	17	8	716	57
9	33	95	6	30	7	14	11	34	26	744

Fig. 54. DenseNet Train Confusion Matrix



Model	Parameters	Test Accuracy	Train Accuracy
SimpleNet	134826	22.7	56.25
Modified	135730	61.35	69.53
ResNet	33660152	64.11	96.9
ResNeXt	388926	64.15	93.97
DenseNet	1124558	65.21	97.28

Fig. 55. Result Table