Homework0: Alohomora

Dhrumil Kotadia
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, Massachusetts
Using One Late Day

I. Phase1: Shake My Boundary

This section focuses on the implementation of the pb-lite boundary detection algorithm, where 'pb' denotes the Probability of Boundary. The algorithm assesses the likelihood of each pixel in the input image belonging to an edge by considering gradients of intensity values, texture and color. The process unfolds through four key steps:

- Generation of a filter bank comprising of Derivative of Gaussian filters, Leung-Malik filters and Gabor filters
- Creation of texton, brightness, and color maps
- Development of texton, brightness, and color gradient maps
- Boundary detection using these maps, along with Sobel and Canny baseline techniques

A. Filter Bank Generation

To capture texture information from images, a diverse set of filters is used to create a filter bank. The three primary types of filters utilized in this phase are:

1) Oriented Derivative of Gaussian (DoG) Filters:
    - Obtained by convolving a Sobel operator on a Gaussian kernel.
    - Includes DoG filters with 2 scales and 16 orientations.
    - Illustrations of Oriented DoG filters are depicted in Figure 1.

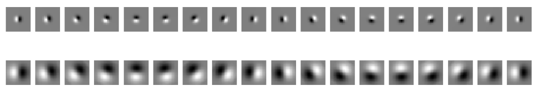


Fig. 1. Derivative of Gaussian Filters

2) Leung-Malik Filters:
    - Comprises 48 filters of multiple scales and orientations.
    - Involves first and second-order derivatives of Gaussians, Laplacian of Gaussian (LoG) filters, and Gaussians.
    - Two Leung-Malik filter banks are generated: LM Small and LM Large.
    - LM Small filters consider scales $\sigma = \{1, \sqrt{2}, 2, 2\sqrt{2}\}$, while LM Large (LML) filters consider $\sigma = \{\sqrt{2}, 2, 2\sqrt{2}, 4\}$.

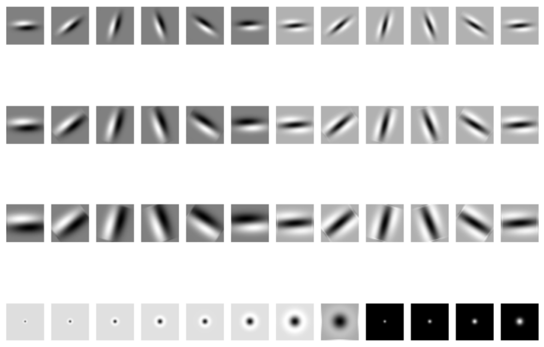


Fig. 2. Leung-Malik Small

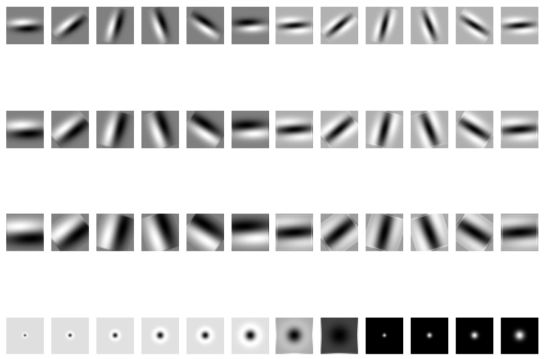


Fig. 3. Leung-Malik Large

    - Filter bank illustrations for LM Small and LM Large are presented in Figures 2 and 3.

3) Gabor Filters:
    - Designed based on the human eye's operation.
    - Features a Gaussian kernel modulated by a sinusoidal wave.
    - The Gabor filter bank created for this experiment is

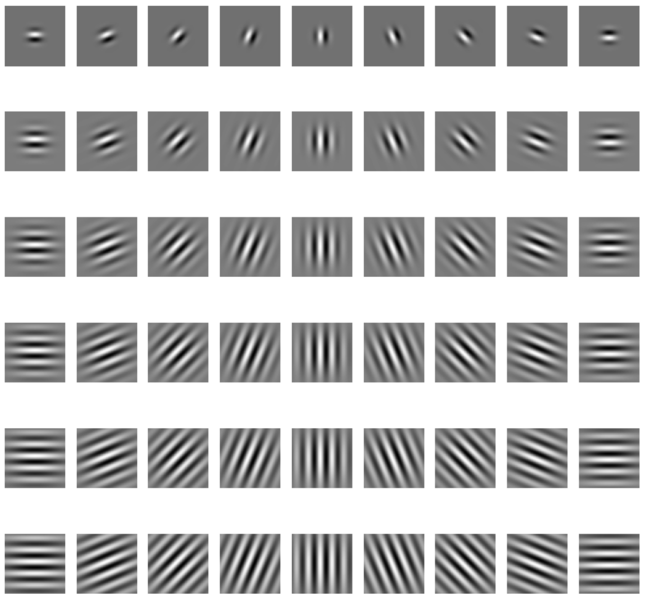


Fig. 4. Gabor Filter

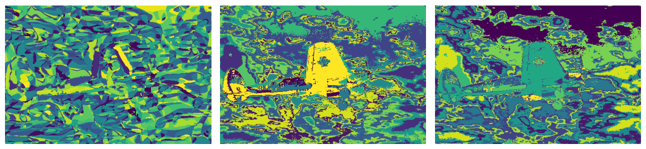
shown in Figure 4.

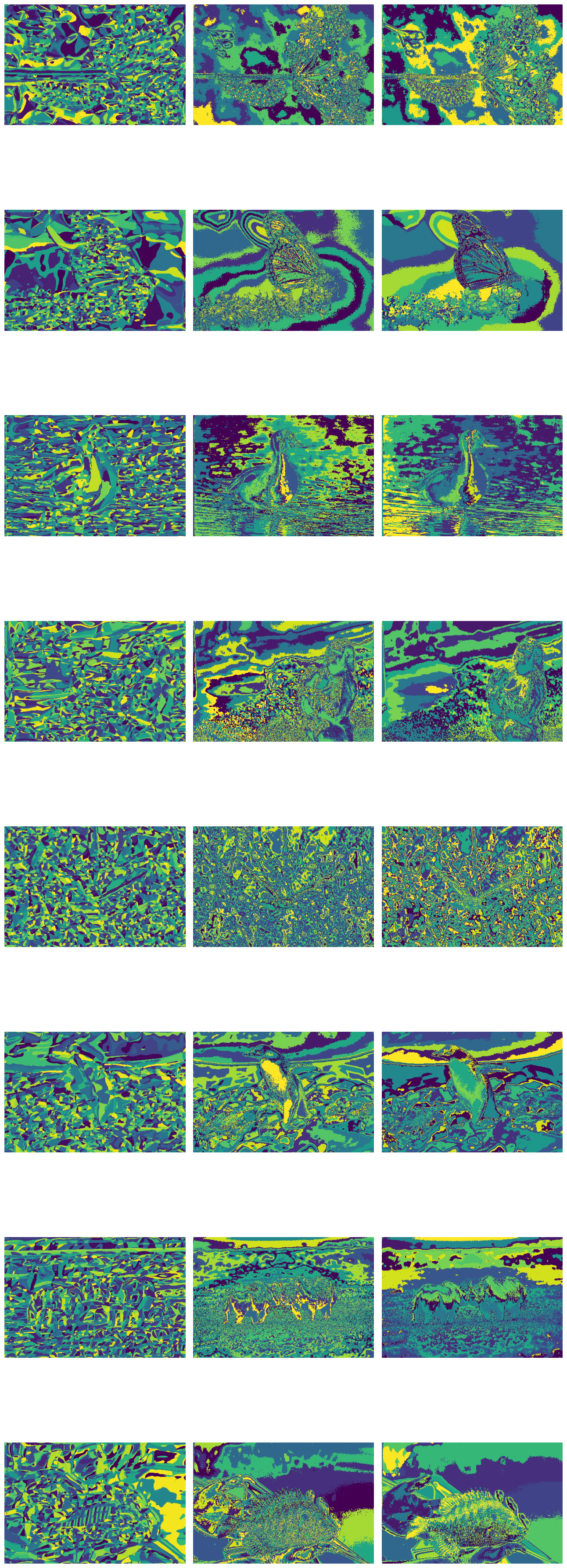
B. Texton, Brightness and Colormap

After the generation of filter banks, each filter from the previously generated banks is applied onto the input images. If the filter banks encompass N filters, the outcome is an N-dimensional vector corresponding to each pixel. Following this, the N-dimensional vectors are replaced with discrete texton IDs. This replacement is achieved by clustering the filter responses for each pixel into K textons using KMeans clustering. Subsequently, each pixel in the image is substituted with the discrete texton ID obtained through KMeans clustering, resulting in the generation of the Texton map (T). The output is a single-channel image with values ranging from [1, 2, ..., K]. For this experiment, a value of 64 was chosen for K.

Similarly, brightness and color maps are generated using analogous methods. To obtain the Brightness map (B), clustering is applied to the brightness or intensity values through KMeans clustering on the grayscale equivalent of the color image. Likewise, Color map (C) is obtained by performing KMeans clustering on the default three-channel color image. A K value of 16 was selected for both brightness and color maps in this experiment.

The Texton, Brightness and Color Maps of all input images observed in figure 5





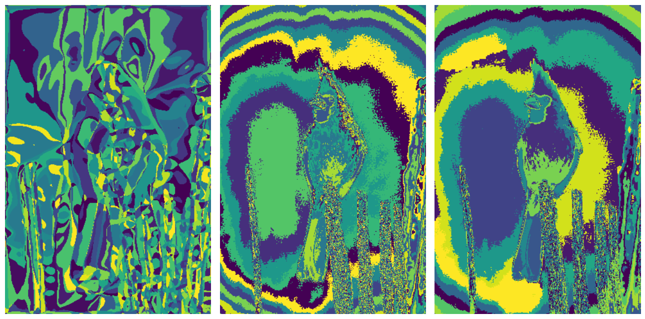


Fig. 5. Texton, Brightness and Color Map for Input Images

C. Texton, Brightness asnd Color Gradients

Gradient maps establish local gradient measurements, indicating the changes in texture, brightness, and color distributions at a specific pixel. To compute Texton, Brightness, and Color gradients, it is essential to calculate variations in values across diverse shapes and sizes. Therefore, an initial step involves generating half-disc masks and employing the chi-square distance metric.

- Half-disc Masks:
  - These masks consist of pairs of binary images representing half-discs.
  - Eight orientations of half-disc masks are created, as illustrated in figure 6.
- Chi-Square Distance:
  - The Chi-Square Distance metric is utilized to measure the dissimilarity between two histograms, specifically, the distinction in distributions within left and right half-disc pairs.
  - The half disc masks created are applied to the Texton,Brightness and Color Maps and their Chi-Square distance is calculated to provide the Texton, Brightness and Color Gradients shown in figure 7

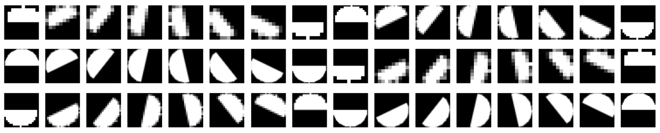
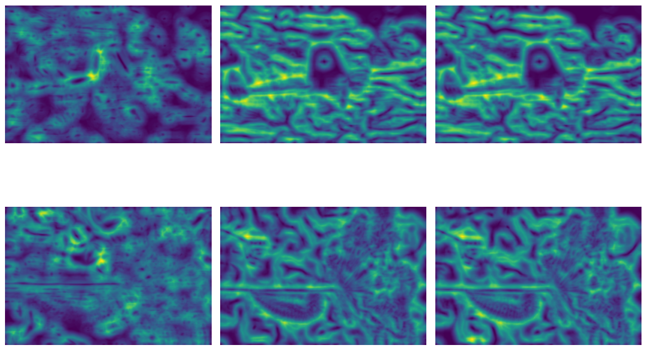


Fig. 6. Generated Half Disc Masks



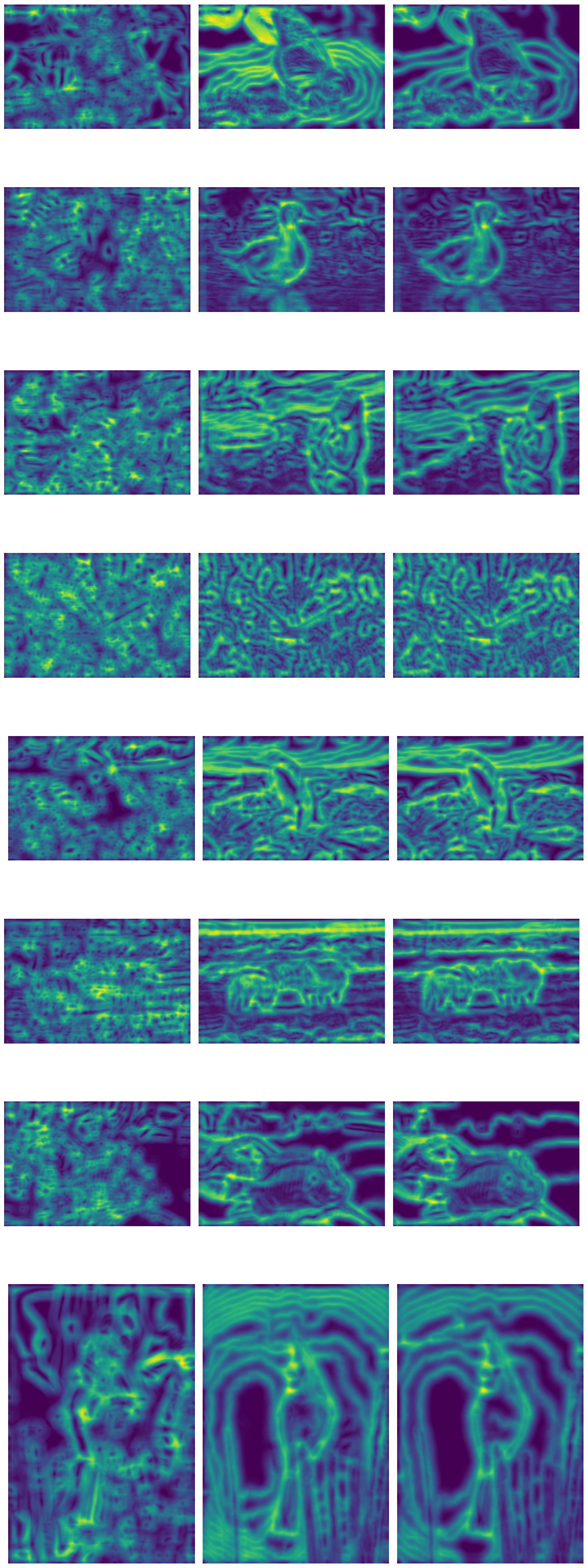


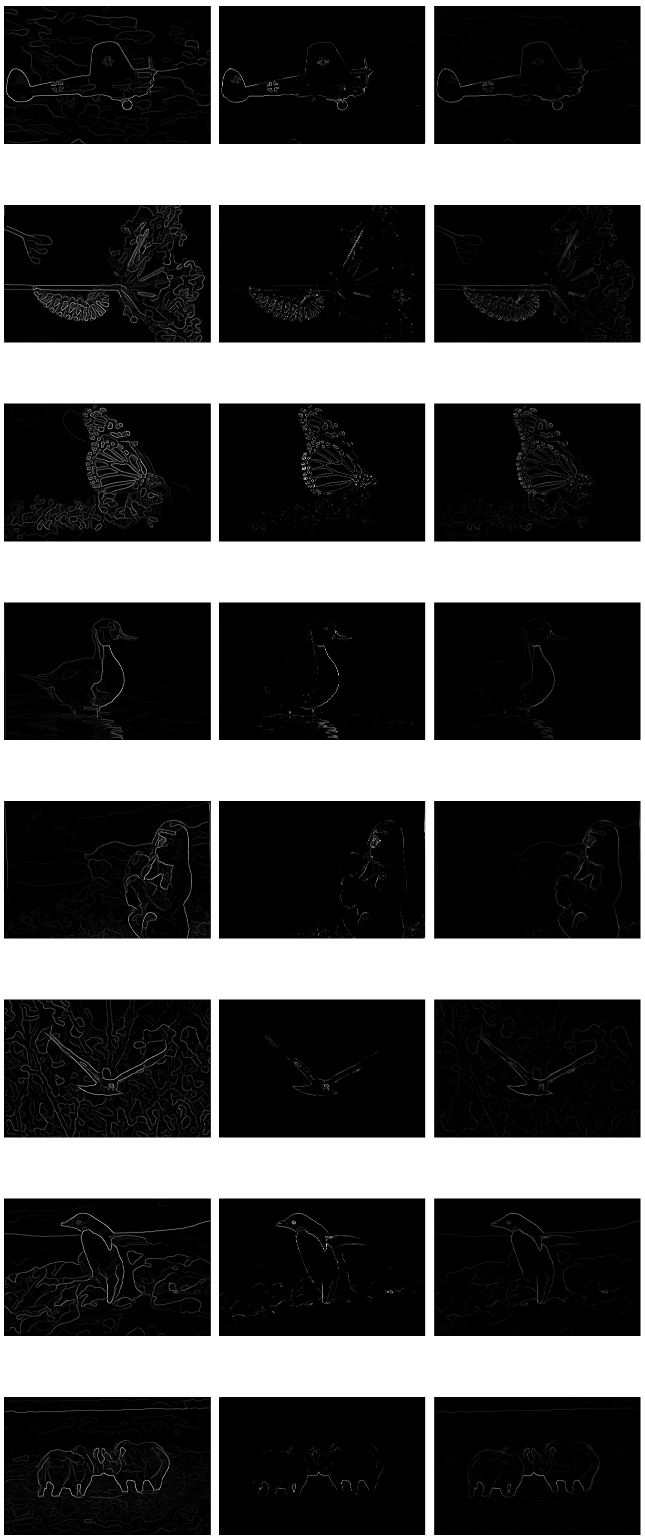
Fig. 7. Texton, Brightness and Color Map for Input Images

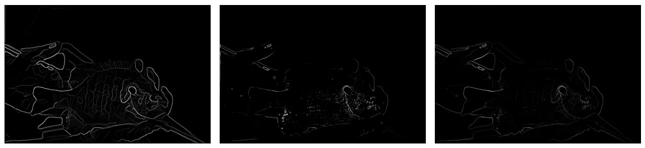
D. PB-Lite Output

The PB-Lite output is calculate based on the equation

$$PBEdges = \frac{T_g + B_g + C_g}{3} \circ (w_1 * cannyPb + w_2 * sobelPb)$$

Here, w_1 and w_2 are chosen to be 0.3 and 0.7 respectively. The output Images generated are shown in figure 8





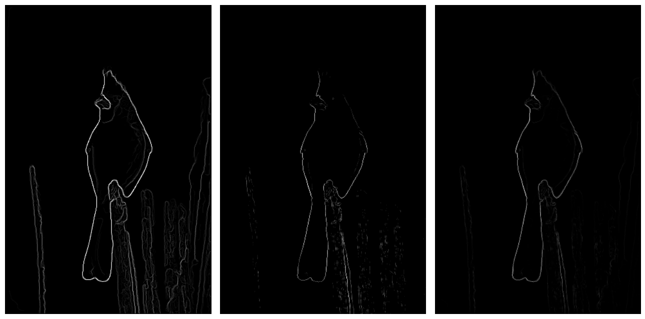


Fig. 8. Canny Baseline, Sobel Baseline and PBLite Output

E. Observation

It can be observed that the PBLite output reduces a lot of false positives compared to Canny Baseline and is able to highlight some minute details that are missed by Sobel baseline. It also has an advantage that there is a lot of room for tuning. The number of filters and the parameters in PBLite provide a lot of robustness since it can be tuned for various scenarios.

II. Phase2: Deep Dive on Deep Learning

A. First Neural Network

For the first neural network, a basic network is implemented which has 2 convolutional layers, 2 pooling layers, a dropout layer and 2 fully connected layers. Both convolutional layers go through ReLU activation followed by Maxpool pooling layers. This is followed by a fully connected layer with ReLU activation. After this we have a 2D dropout layer with probability 0.25 followed by another fully connected layer. This makes up the entire network. Batch size of 32 was selected for training and AdamW optimiser was chosen. The AdamW optimiser has learning rate of 0.01 and weight decay of 0.0001. This is run for 20 epochs and the result is shown in the figure 9. The final accuracy obtained while training is 41.85%. The observation here is that the accuracy does not increase. It waivers slightly up and down. The reason behind this can be high learning rate.. The testing accuracy for this model is also similar (41.06%). The number of parameters in this model are 8.

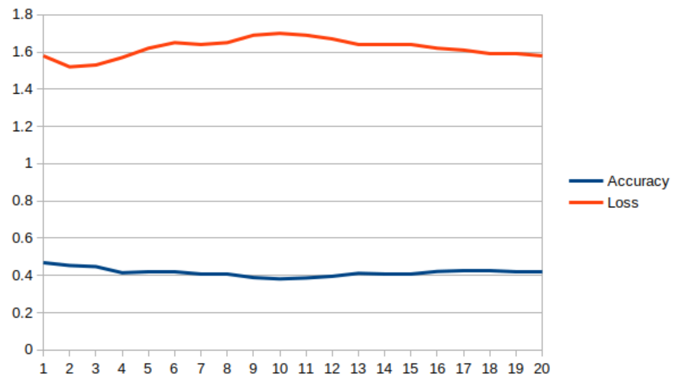


Fig. 9. Basic Network with learning rate 0.01 and batch size 32

The confusion matrix of training this model is shown in figure 10

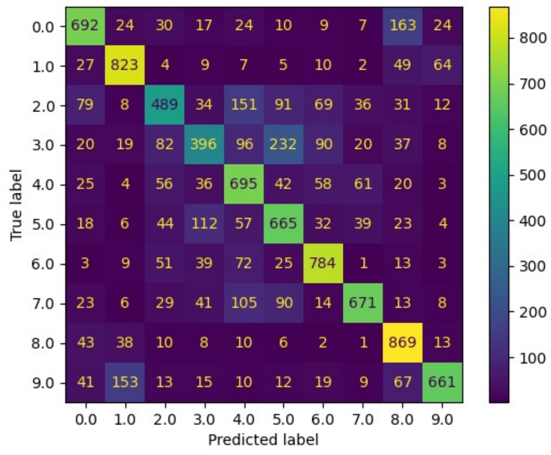


Fig. 10. Basic Network training confusion matrix

The confusion matrix of testing this model is shown in figure 11

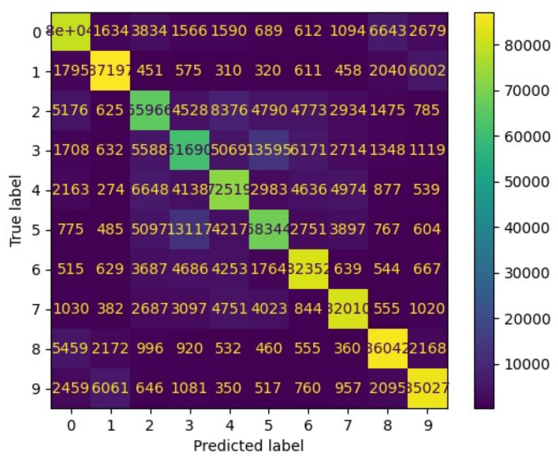


Fig. 11. Basic Network testing confusion matrix

B. *Updated Basic Network*

The model was trained again with different parameters. Now, the learning rate was kept variable where it decreases with increasing number of epochs. LR_Scheduler was used to have a decay with step size of 5 and gamma 0.1. Batch size was updated to 128. The resulting plot can be seen in figure 12.

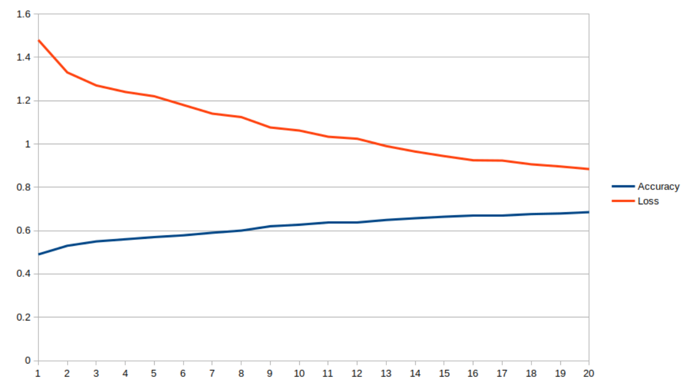


Fig. 12. Basic Network with learning rate 0.001 and batch size 128

Here it is evident that the accuracy increases over time and the loss decreases. The confusion matrix of training this model is shown in figure 13

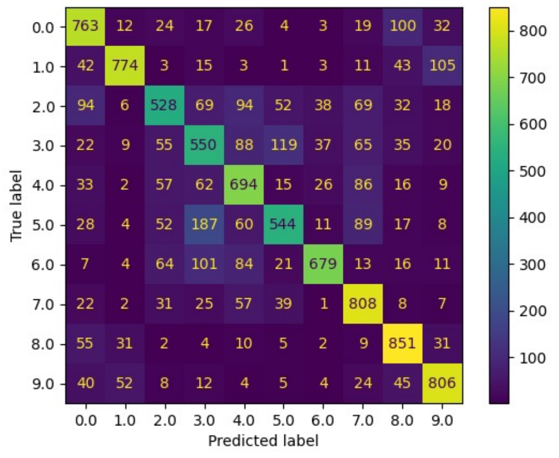


Fig. 13. Updated Network training confusion matrix

The confusion matrix of testing this model is shown in figure 14

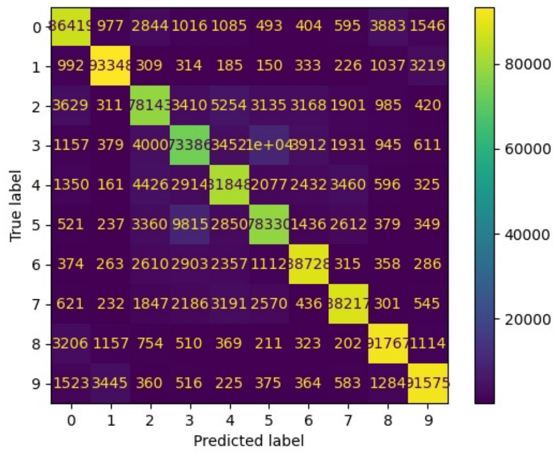


Fig. 14. Updated Network testing confusion matrix

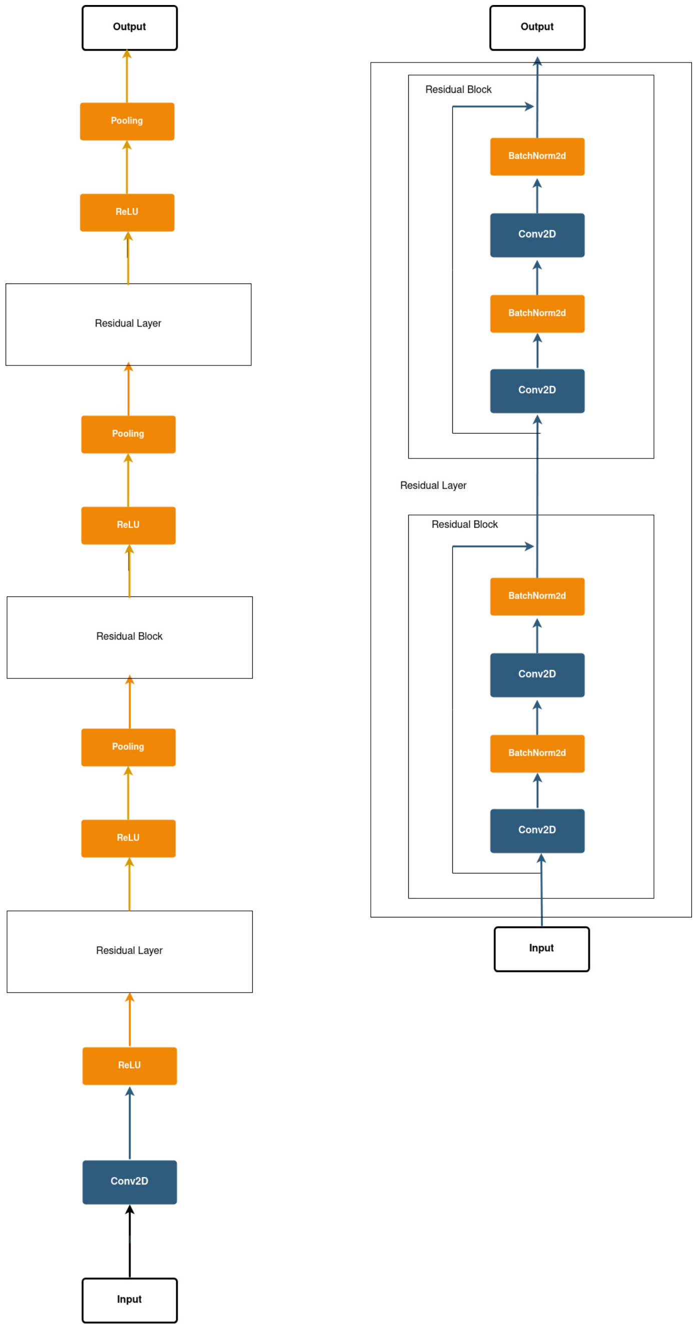


Fig. 15. ResNet Model

C. Resnet

ResNet is an architecture that uses the idea of residual blocks to overcome the problems of diminishing performance in very deep networks. After each residual block, the input of the residual block is added to the output of the residual block. This connection allows the network to skip one or more layers during training. The model of the implemented ResNet is shown in figure 17

The learning rate is set to 0.001 and the batch size is 128. The model is trained for 5 epochs. The plot is visible in figure 16

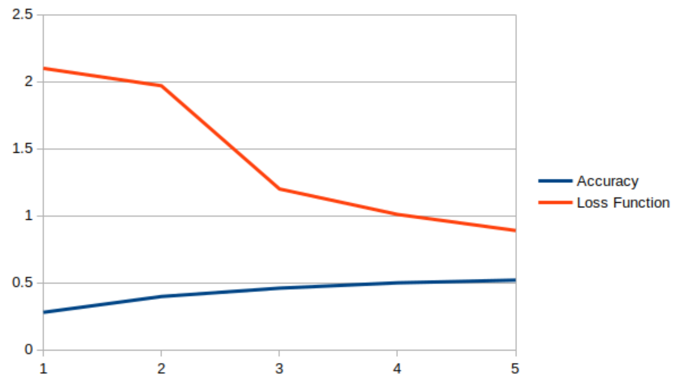


Fig. 16. ResNet Training Accuracy and Loss

D. ResNext

ResNext is an architecture that is an extension of Resnet. It uses the residual blocvks just like the resnet architecture. The only difference is cardinality. ResNext has a lot of layers inside the residual block parallelly training before they are all concatenated together and then added to the input just like the Resnet architecture. The use of cardinality allows ResNeXt to achieve competitive or better results than traditional ResNet architectures.

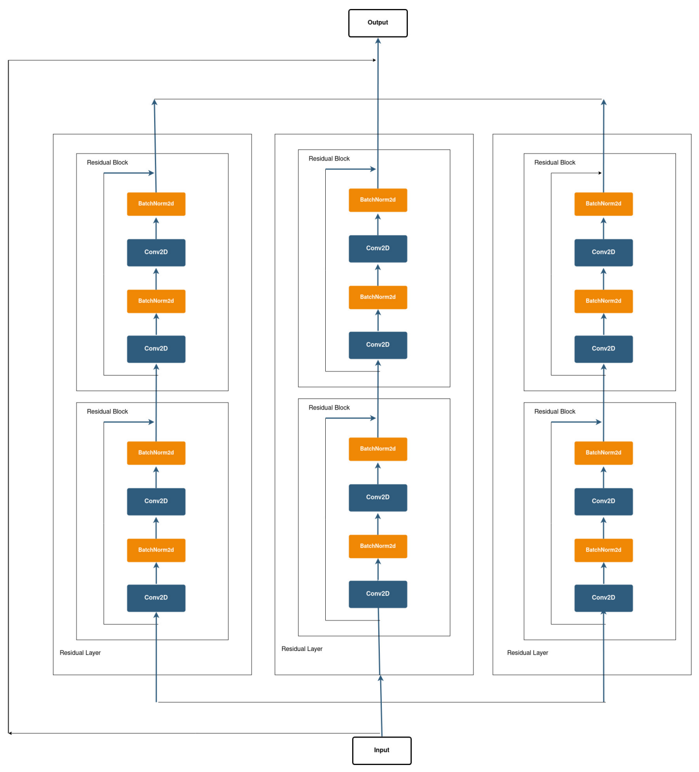


Fig. 17. ResNext Model

The learning rate is set to 0.001 and the batch size is 128. The model is trained for 5 epochs. The graph is visible in figure 18

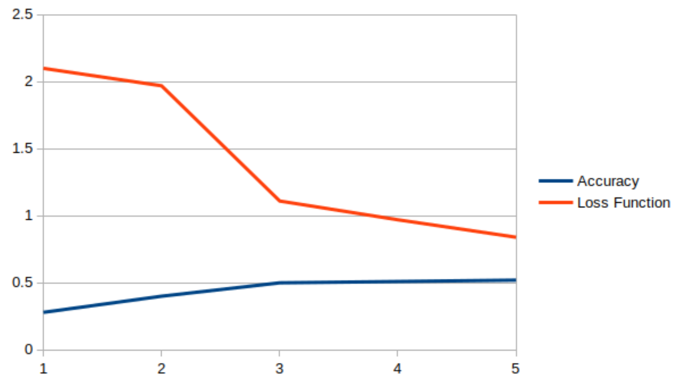


Fig. 18. ResNet Training Accuracy and Loss

E. Densenet

Densenet is an architecture where concatenation occurs after every layer in a dense block. If there are l previous layers in a dense block, then the output of the current layer is concatenated with l-1 outputs of all previous layers. In the current implementation, 2 such dense blocks are implemented each having 3 layers with a transitional layer after each dense block. A growth rate of 12 is used in the current implementation. It has the following layers: First, a convolution layer, then a dense block which has 3 convolutional layers. Concatenation is done after every layer in the densenet block. After that there is a transitional layer which comprises of maxpool and a convolutional layer. After that, another dense block which has 3 more convolutional layers with concatenation after each layer. Then there is a maxpool and a convolutional layer, followed by a fully connected layer which gives the output. The learning rate is set to 0.001 and the batch size is 128. The model is trained for 5 epochs. The Graph for DenseNet is as shown in figure 20

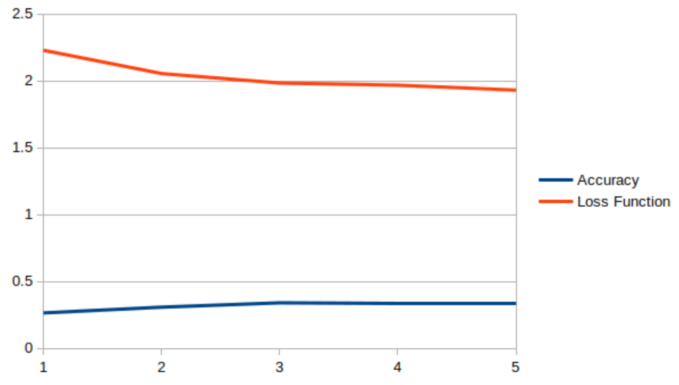


Fig. 20. DenseNet Accuracy and Loss

F. Analysis

The basic network really underperforms but improves when the parameters are changed in the updated Basic network. The ResNet architecture gets high accuracy even in low epochs and performs much better than basic architecture. ResNext also perform much better than the basic architecture but DenseNet does not perform as well. The cause behind this can be the fact that it might not be implemented and tuned properly.

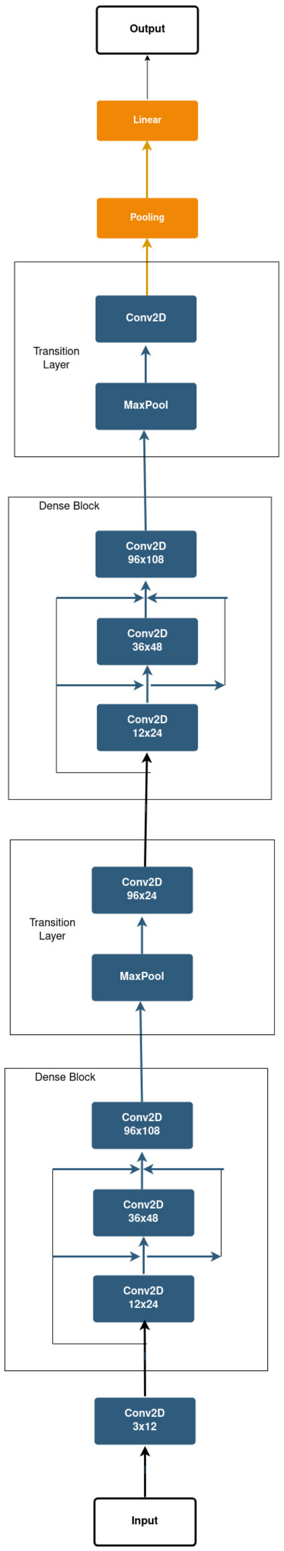


Fig. 19. DenseNet Model