Homework 0 - Alohomora
USING 1 LATE DAY

Harsh Verma
Robotice Engineering
Worcester Polytechnic Institute
hverma@wpi.edu

I. Introduction

Phase 1 involves generating a superior method of boundary detection, the pb-lite algorithm. We start with generating various filter banks including oriented DoG, Leung-Malik, Gabor and HD masks.

We use basic functionalities like generating Gaussian Filters, and convolving filters with each other to gain these filter banks. A brief description of how each filter bank is generated is as follows:

A. Oriented Derivative of Gaussian

DoG: Derivative of Gaussian filter bank is generated by convolving Gaussian filters with Sobel filters. Sobel filter acts like a differentiation operation in this case. We used a pre-defined Sobel filter to generate this filter bank. Two scales, 2 and 3 are used to generate the image.

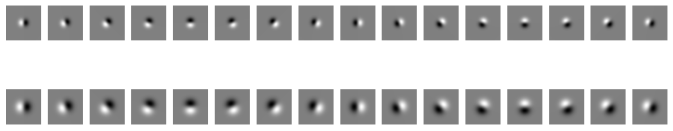


Fig. 1. DoG filter bank

B. Leung-Malik filters

Leung-Malik Filter bank consists of various types of filters-The first and second derivative of Gaussian Filters with elongated scales, Laplacian of Gaussian filter, and normal Gaussian filters. All these filters amount to 48, and have two versions-small and large, based on the scale they follow.

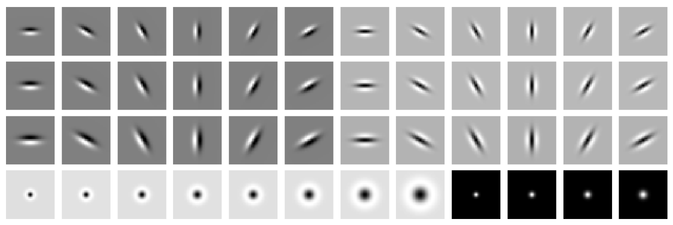


Fig. 2. LMS filter bank

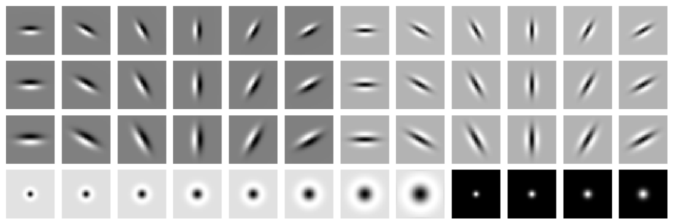


Fig. 3. LML filter bank

C. Gabor Filters

The Gabor Filter is an interesting filter made up of Sinusoidal pattern being rotated according to the orientation, and convolved with a Gaussian filter kernel.

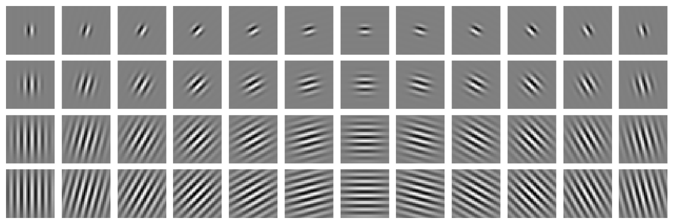


Fig. 4. Gabor filter bank

D. Half Disc Masks

Half Disc masks are a set of different orientations of pairs of semi-circular shaped filters. These filters help map the changes in brightness of the images.

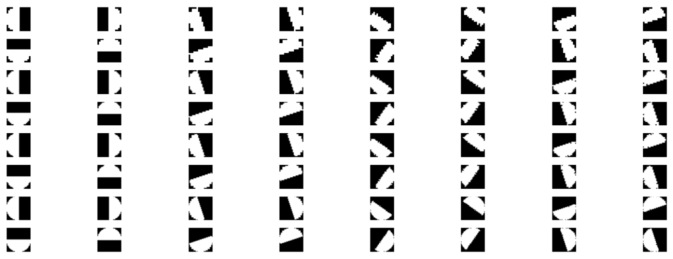


Fig. 5. Half Disc mask filter bank

E. *Maps*

These maps are representations of various patterns found in images. The more advanced applications of these maps are in Segmentation and Recognition. I used Kmeans from the sklearn.cluster library in python to create these maps.

II. Phase 2 - Deep Dive on Deep Learning

This is a classification problem solved using deep learning. We use the popular CIFAR-10 dataset having 50000 training images and 10000 test images. There are 10 classes in the dataset, hence the final layer of all of our Neural networks would be 10.

All the networks have been run for 10 epochs, have a batch size of 32 and learning rate of 0.001

A. *CIFAR10Model() - simple model (Accuracy 26.1%)*

Inspired by the official Pytorch tutorials, I created this simple neural network with the features and parameters in the summary diagram.

B. *ImprovedModel - Improved performance model (Accuracy 40.4%)*

To create this model, I started with 4 conv2d layers and 3 linear layers at the end. Between each of the convolution layer, I used ReLU activation. As suggested, I also normalised the outputs after the third convolution layer.

C. *ResNet (Accuracy 47.4%)*

Resnet architecture inspired by 'ResNet9' was implemented.

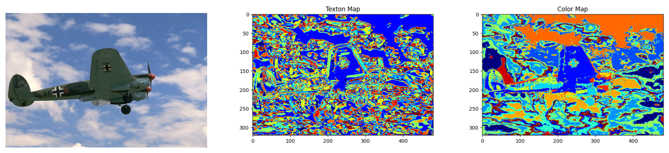


Fig. 6. Original image, Texton Map, Color Map

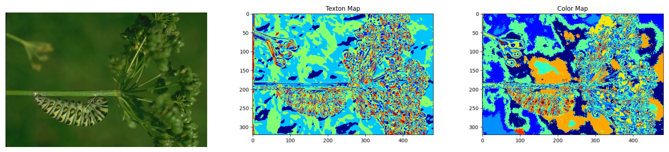


Fig. 7. Original image, Texton Map, Color Map

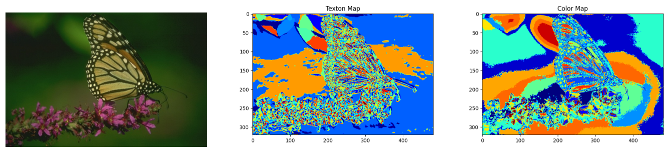


Fig. 8. Original image, Texton Map, Color Map

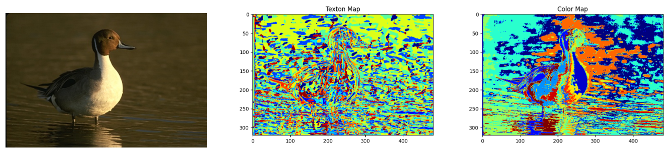


Fig. 9. Original image, Texton Map, Color Map

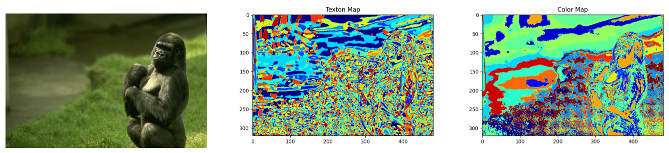


Fig. 10. Original image, Texton Map, Color Map

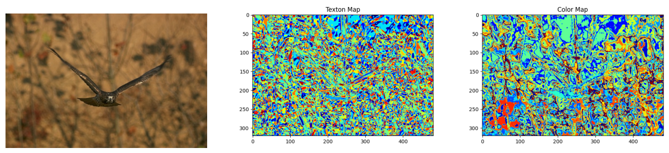


Fig. 11. Original image, Texton Map, Color Map

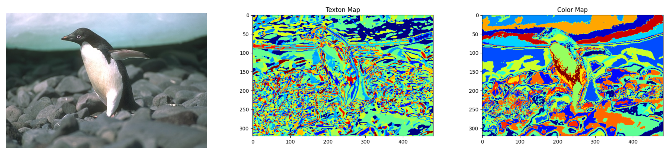


Fig. 12. Original image, Texton Map, Color Map

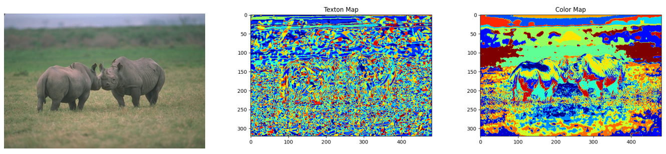


Fig. 13. Original image, Texton Map, Color Map

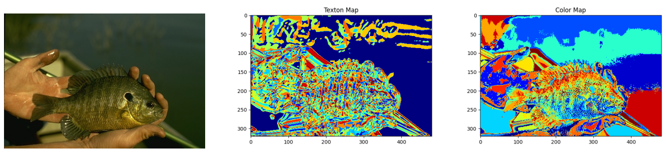


Fig. 14. Original image, Texton Map, Color Map

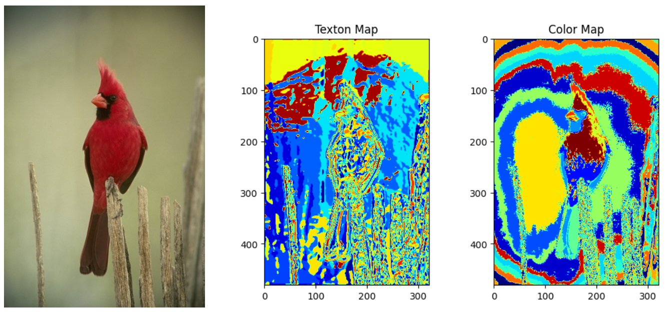
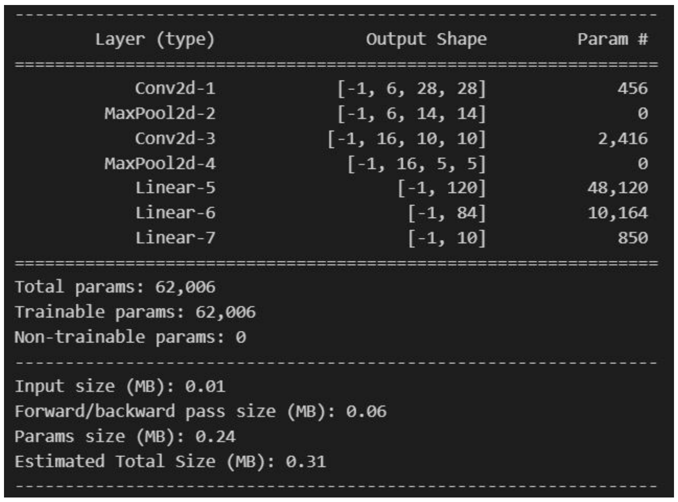


Fig. 15. Original image, Texton Map, Color Map



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [-1, 6, 28, 28]             456
         MaxPool2d-2            [-1, 6, 14, 14]               0
            Conv2d-3           [-1, 16, 10, 10]           2,416
         MaxPool2d-4             [-1, 16, 5, 5]               0
            Linear-5                  [-1, 120]          48,120
            Linear-6                   [-1, 84]          10,164
            Linear-7                   [-1, 10]             850
================================================================
Total params: 62,006
Trainable params: 62,006
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 0.06
Params size (MB): 0.24
Estimated Total Size (MB): 0.31
----------------------------------------------------------------
```

Fig. 16. Summary of the simple model

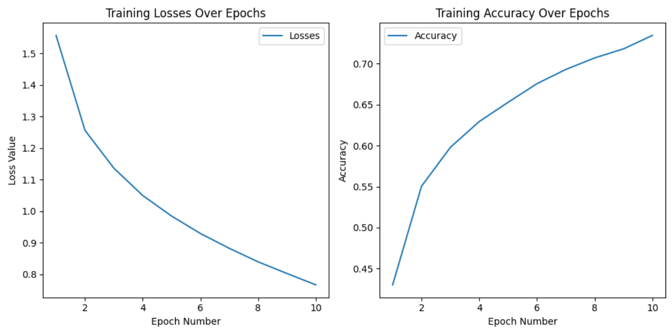
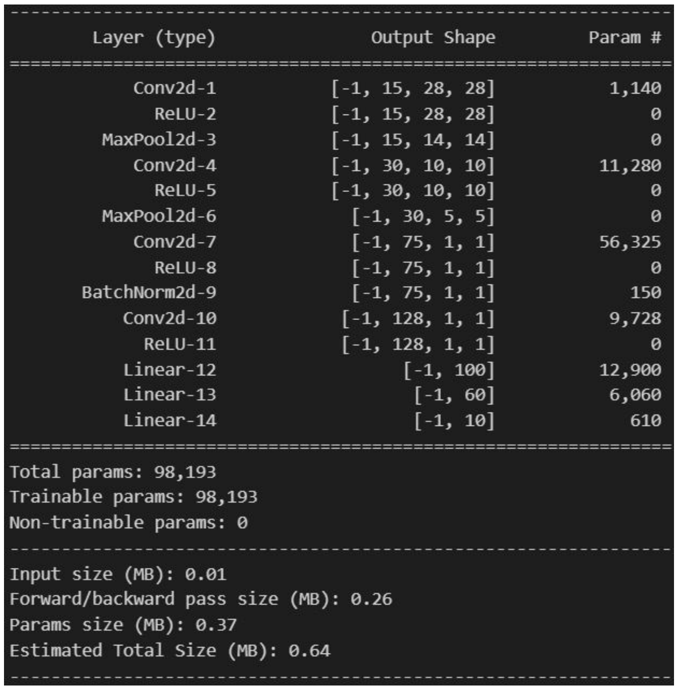


Fig. 17. Performance of the simple model



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 15, 28, 28]           1,140
              ReLU-2           [-1, 15, 28, 28]               0
         MaxPool2d-3           [-1, 15, 14, 14]               0
            Conv2d-4           [-1, 30, 10, 10]          11,280
              ReLU-5           [-1, 30, 10, 10]               0
         MaxPool2d-6             [-1, 30, 5, 5]               0
            Conv2d-7             [-1, 75, 1, 1]          56,325
              ReLU-8             [-1, 75, 1, 1]               0
       BatchNorm2d-9             [-1, 75, 1, 1]             150
           Conv2d-10            [-1, 128, 1, 1]           9,728
             ReLU-11            [-1, 128, 1, 1]               0
           Linear-12                  [-1, 100]          12,900
           Linear-13                   [-1, 60]           6,060
           Linear-14                   [-1, 10]             610
================================================================
Total params: 98,193
Trainable params: 98,193
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 0.26
Params size (MB): 0.37
Estimated Total Size (MB): 0.64
----------------------------------------------------------------
```

Fig. 18. Summary of the ImprovedModel

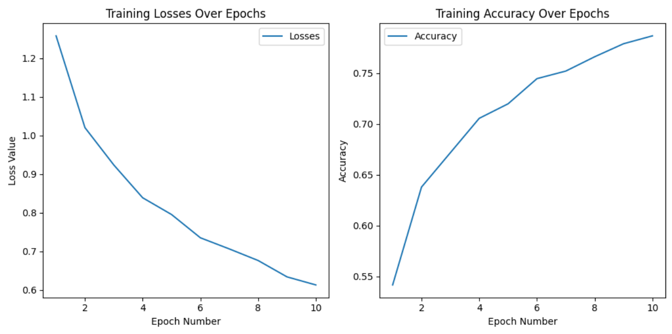


Fig. 19. ImprovedModel Performance

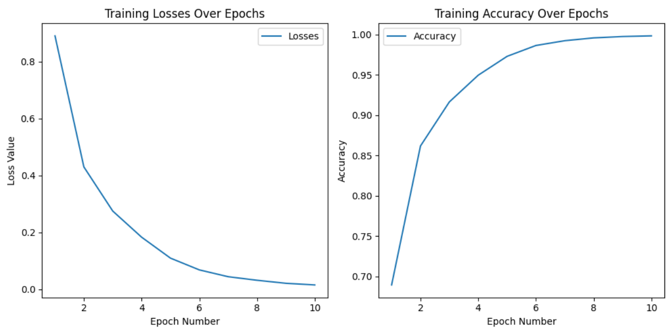


Fig. 20. Resnet Performance

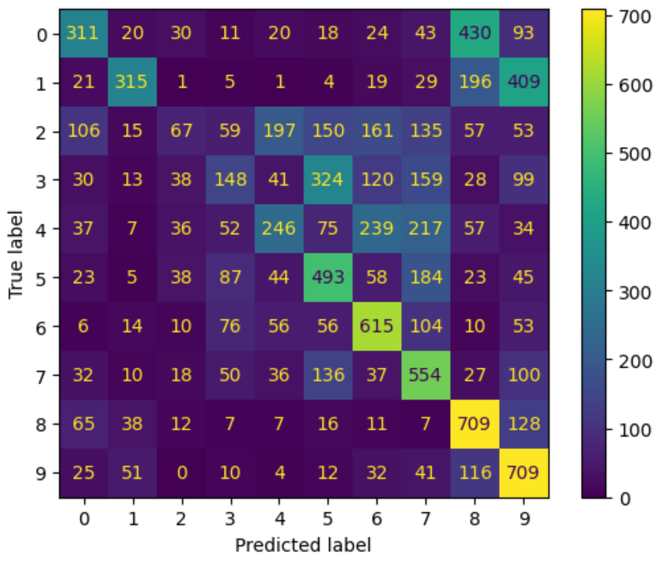


Fig. 21. Improved Model Confusion Matrix

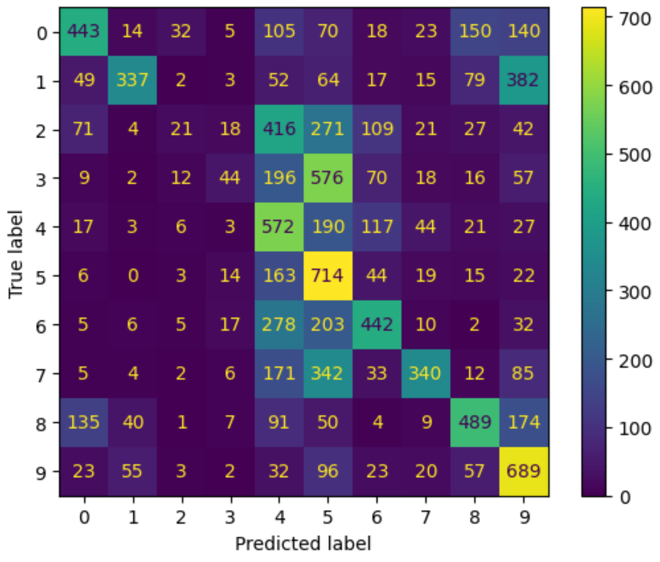


Fig. 22. Resnet Confusion Matrix