HW0 : Alohomora

Anuj Jagetia
M.S. Robotics Engineering
email : ajagetia@wpi.edu

I. Phase 1 : Shake My Boundary

This goal of this section is to apply the pb-lite border (a lite version of probability of border). It finds boundaries by assessing brightness, color, and texture information across many scales (different sizes of objects/image) in order to detect edges which produces a probability of border for each pixel. There are four steps to this process:

1) Defining different Filter Banks
2) Computing Texton, Brightness and Color Maps
3) Finding gradients for above maps
4) Combining the outputs of Sobel and Canny Baselines

A. Filter Banks

Filtering the image using a collection of filter banks is the initial stage in the pb lite border detection process. For this, we created three distinct sets of filter banks.

1) Oriented Difference of Gaussian: This a Convolution of Sobel operator to a gaussian kernel and rotated in 16 different orientations within 0 to 360 degrees and 2 scales.

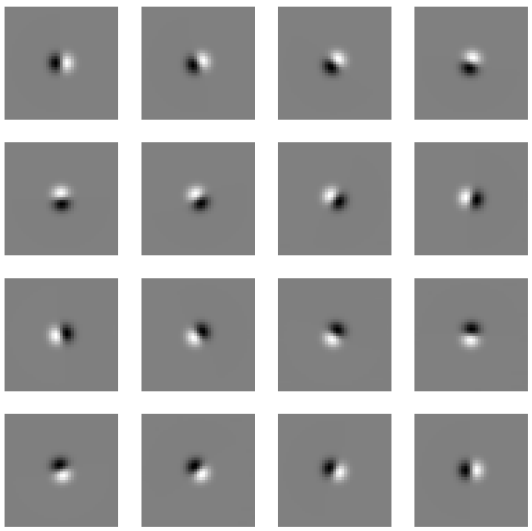


Fig. 1. *Oriented DoG Filter Bank*

2) Leung-Malik Filters: It is a filter with multiple scales and orientations. In total 48 filters comprises of Gaussian, Laplacian, First and Second order derivatives of Gaussians. LM Filter is divided in two categories Small and Large LM filters, the difference between them is they have different scales.

3) Gabor Filters: The basis for the construction of gabor filters are what we found in the human visual system. A sinusoidal plane wave modulates a gaussian kernel function, which is called a gabor filter.

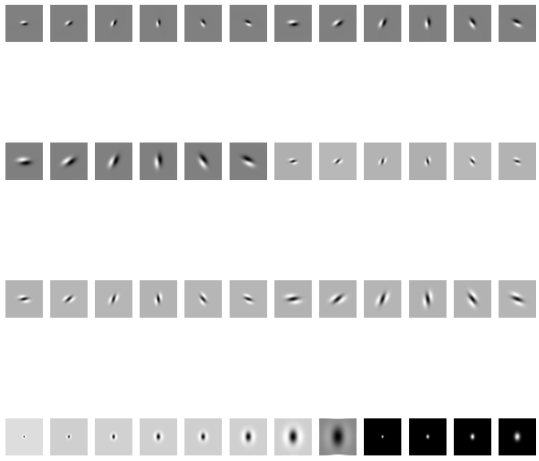


Fig. 2. *LMS Filter Bank*

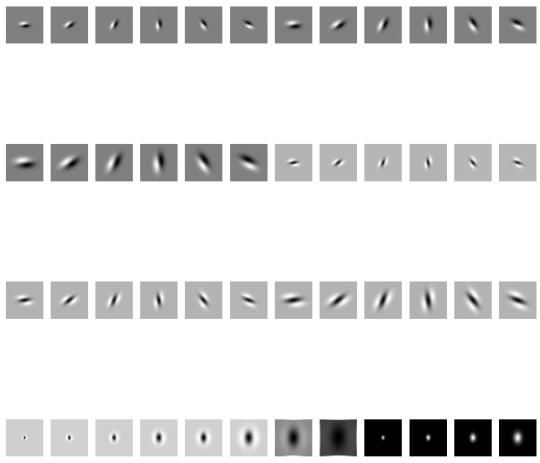


Fig. 3. *LML Filter Bank*

B. Texton, Brightness, Color Maps

Texton maps are made by stack of the outputs which is produced after applying each filters to the image. Now we assign a separate texton ID to each pixel and group the pixels with similar texture attributes, which is done by KMeans clustering approach where the related pixel values are assigned to one cluster. In this instance, each pixel is classified using 64 cluster

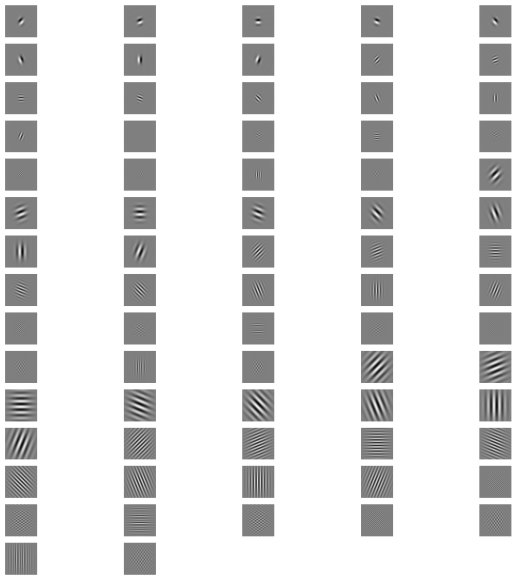
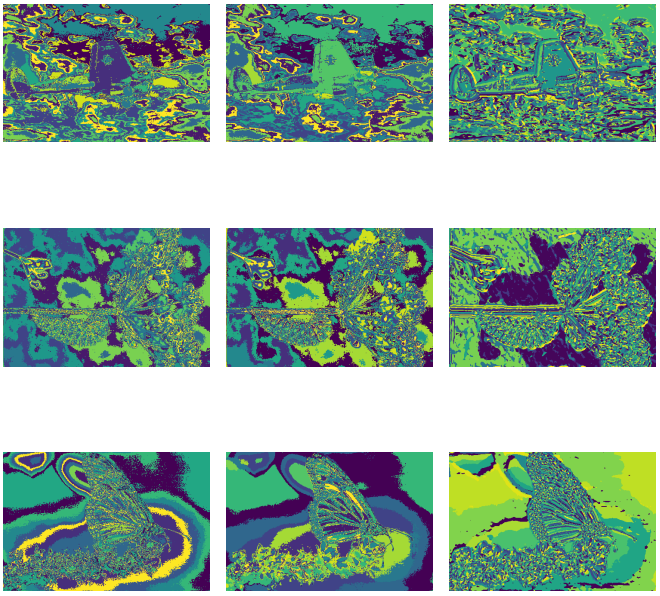


Fig. 4. *Gabor Filter Bank*



centers. As the number of cluster increases the texture details also increase and vice versa.

The texture properties of the image are provided by the texton map. A vector of filter responses centered on each pixel is produced when a single filter—168 in this example—is applied to the image to create Texton maps. At each pixel, we currently have 168 filter responses. A discrete texton ID is then assigned to each pixel after the pixels with similar texture properties are grouped using the K-Means clustering method with a number of clusters equal to 64. The intensity and color values for each pixel are similarly encoded using the brightness and color maps, respectively.

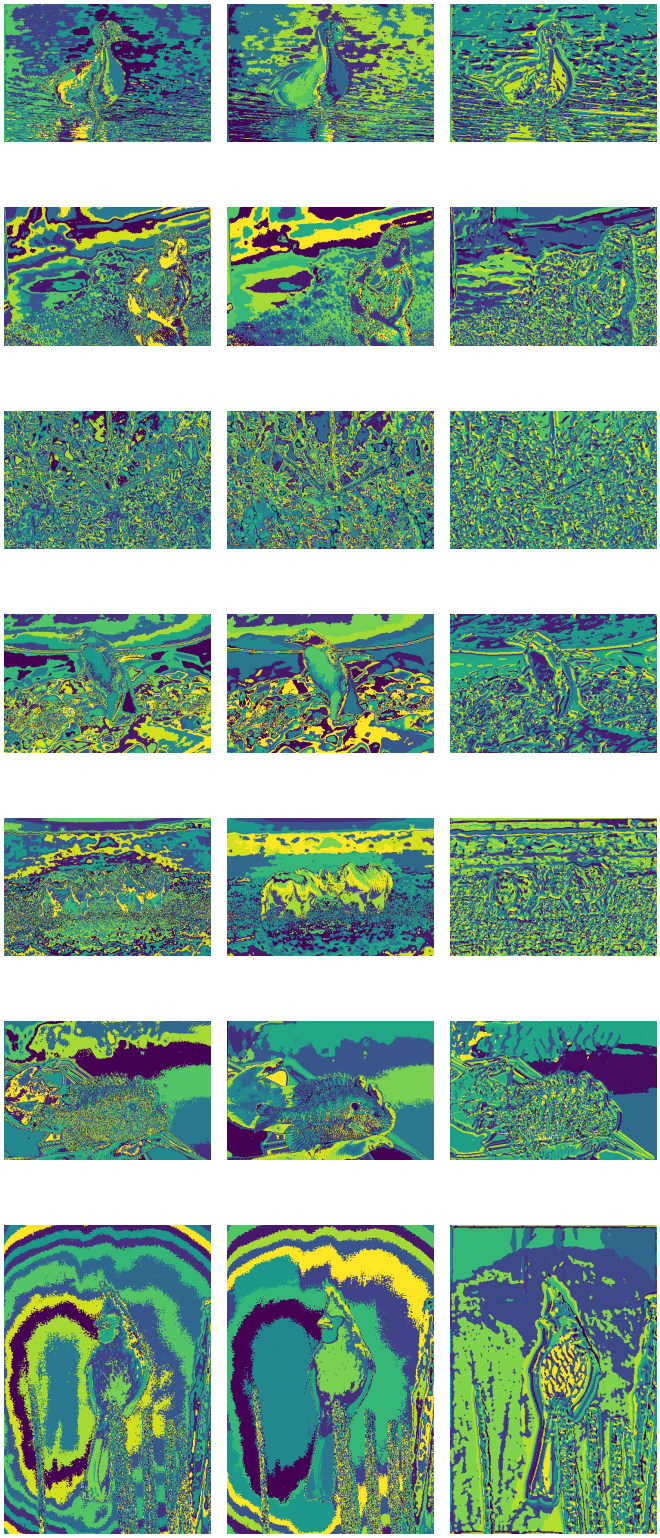


Fig. 5. Texton, Brightness, Color Maps for Images

C. Texton, Brightness, Color Gradients

Gradient maps are useful in order to understand the pixel neighbourhoods where change in texture, intensity and color properties was happening. We must compute value differences

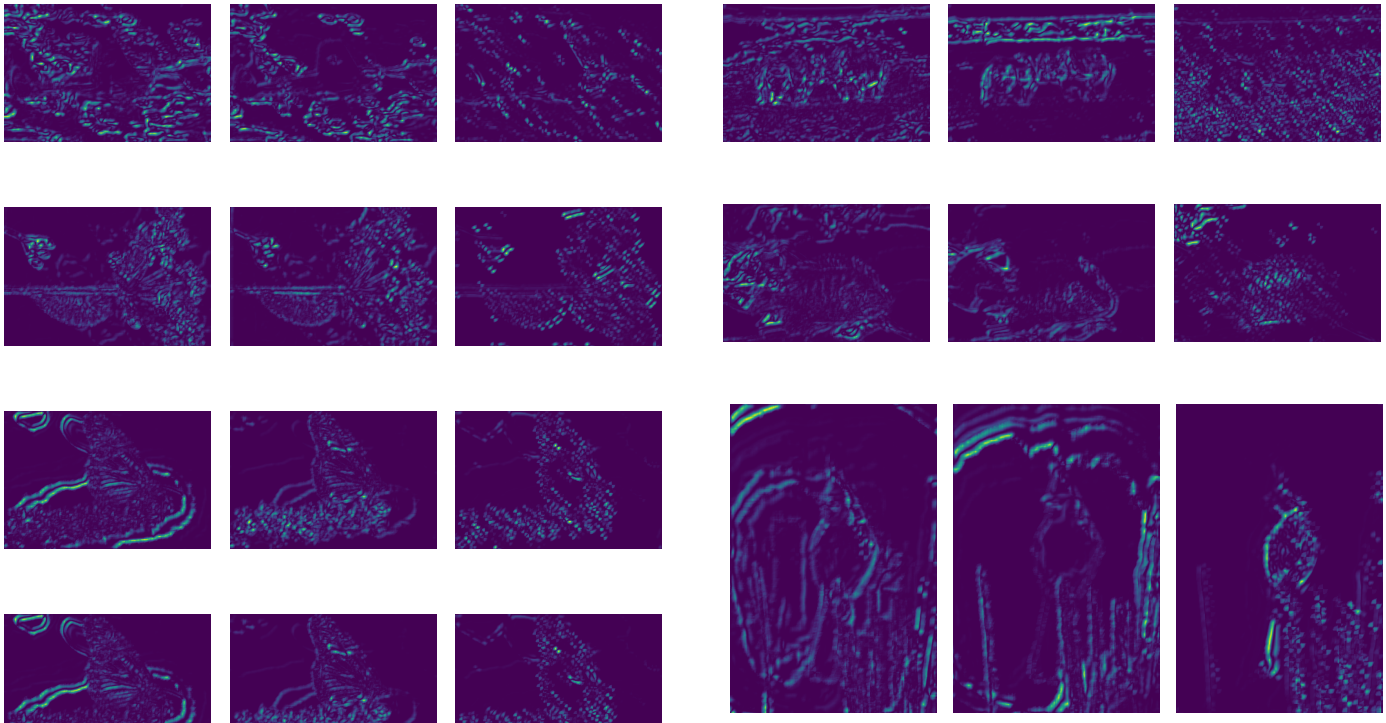


Fig. 6. Texton, Brightness, Color Gradients for Images

across various shapes and sizes in order to compute gradients for texton, brightness, and color. As a result, the chi-square distance and half disk mask are first constructed.

1) Half-disc masks: The pairs of binary images of hard-disc is known as hard disk mask. These discs are generated for different orientations.

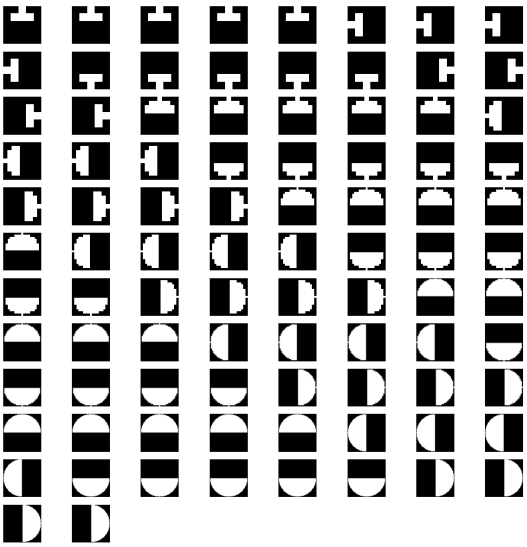


Fig. 7. *Half-Disc Filter Bank*

2) Chi-Distance: The Chi-Distance is used to calculate the difference between the distributions in left and right half-disc pairs.

D. Boundary Detection

To create the pb-lite output, we have all of the gradient maps for the input pictures. But before that we read the Sobel and Canny baselines for these input images in order to do so.

Fig. 8. Canny, Sobel and Pb-Lite Output of Images

E. Results for Phase 1

Based on the findings, we can conclude that Pb-lite performs better at canceling noise in the Canny output; however, in terms of edge detection, Sobel perform better. Pb-lite can control the intensity of texture, brightness and color details, which can be obtained by choosing optimized weights. Additionally, the filter bank is crucial as it allows for the improvisation of responses using the best scales and kernel sizes.

II. Phase 2 : Deep Dive on Deep Learning

The goal of this phase was to implement several neural network architectures for the purpose of performing classification operations on the CIFAR-10 dataset, which consists of 10,000 testing and 50,000 training 32x32 images from 10 classes.

Unfortunately, I was only able to train one model i.e., Basic Neural Network. But we were supposed to develope 3 more models in the experiment, which were Resnet, ResNeXt and DenseNet

A. Basic Neural Network

To classify the images into ten classes, the basic architecture consists of 2 linear layers and 3 convolution layers with max pooling. SGD(Stochastic Gradient Descent) is the employed optimizer, with a learning rate of 0.0001. The network is structured with the following parameters.

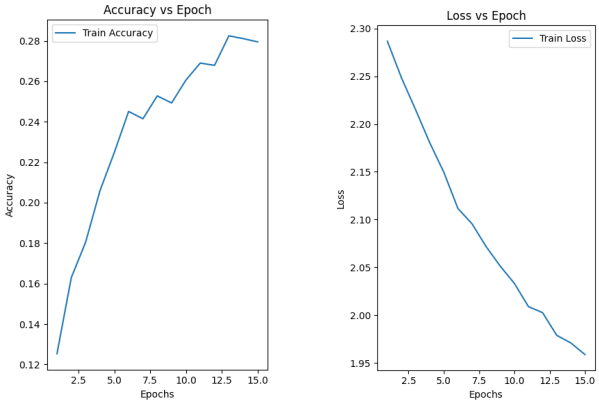
1) There are 15 epochs.
2) There are 30 mini batches.
3) SGD is the optimizer.

In the absence of standardization and data augmentation, 27.57 % accuracy was attained on the testing set.

B. Improved Neural Network

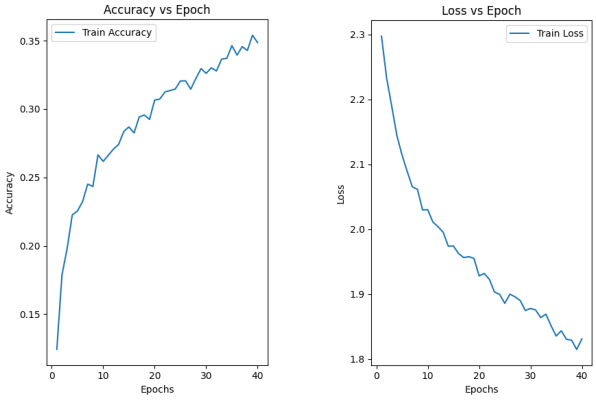
To enhance the performance of our neural network, we can make adjustments such as changing the scale, tuning the decay rate, decreasing batch size, adding batch normalization between layers, and modifying hyperparameters like the number of layers or neurons in a particular layer.

In the initial attempt, we achieved an accuracy of approximately 28 % and by simply adjusting the batch size and the number of epochs per batch, we observed a 7 % improvement without introducing additional layers or neurons. It's important



(a) Training Accuracy (b) Training Loss

Fig. 9. Basic Neural Network



(a) Training Accuracy (b) Training Loss

Fig. 10. Improved Neural Network

to note that as accuracy increases, the training time of the model also tends to increase.

For the improved basic architecture, we utilized the following parameters:

1) 40 epochs
2) 15 mini-batches
3) Stochastic Gradient Descent (SGD) optimizer

With this we achieved a total accuracy of around 35

III. ResNet, DenseNet

I can not find a way to properly deploy a ResNet and DenseNet Network, I had lots of Bugs in the code therefore could not train the model but I am trying to find a way to remove the problems and train a ResNet model.