Homework 0: Alohamora!

Irakli Grigolia
RBE549: Computer Vision
Worcester Polytechnic Institute, Worcester 01609
igrigolia@wpi.edu
Using 2 Late Days

***Abstract*—This paper is separated into two parts. In the first part, the we discuss the various components of Phase 1, which includes an overview of the process, filter banks, texton, brightness and color maps, as well as their gradients and the final output of the pb-lite algorithm for a test image. In the second part, we delve deeper into the topic of deep learning by training and analyzing a convolutional neural network (CNN) for image classification.**

I. PHASE 1: SHAKE MY BOUNDARY

The main requirement was to run pb-lite edge detection, a condensed version of the pb boundary detection technique, on a collection of photos and contrast the results with those from more established edge-detection algorithms like Sobel and Canny. The Pb-lite method employs more picture discontinuities than just intensity discontinuities. It examines a number of picture characteristics, including texture, color, and intensity discontinuities, and then produces a probabilistic output that categorizes each pixel as being a component of an edge or not. Four phases make up the algorithm: 1) Create a filter bank 2) Generate a texture, brightness, and color map 3) Make gradient maps 4). Combine from the features with a baseline method.

A. Filter Banks

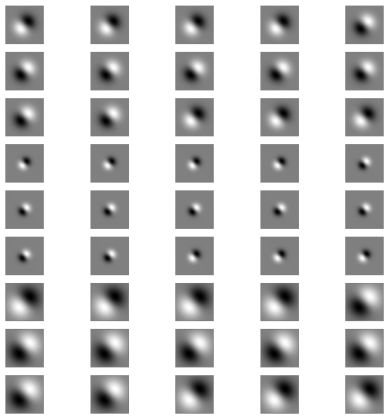


Fig. 1: DoG Filter Bank

A filter-bank in computer vision refers to a set of filters applied to an image or video stream to extract specific features or characteristics. These filters can be designed to detect edges, textures, colors, or other visual elements that are important for image analysis and recognition tasks. The output of a filter-bank is typically a set of filtered images, each highlighting a different aspect of the original image. These filtered images can then be used as inputs to other computer vision algorithms, such as object detection or image classification, to improve their performance and accuracy.

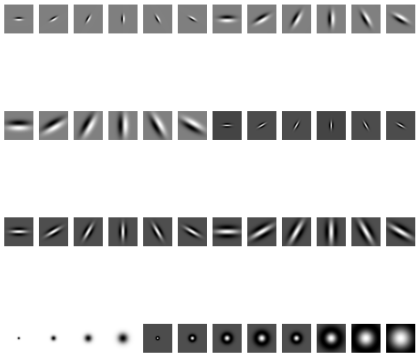


Fig. 2: LM Filter Bank

We used combinations of 3 different filter banks : Originated DoG, Leung-Malik, and Gabor Filters. We can see them on Figures 1,2, and 3 respectively.

B. Texton, Brightness, and Color Maps

Textons are a concept used in image processing and computer vision to describe the basic elements of texture in an image. They are created by applying a filter bank to an image, which separates the image into different frequency bands. After that we perform dimensionality reduction that gives N dimension vector for each pixel through a process, resulting in a matrix of (R,W,N). To simplify this N dimension vector, a K-Means clustering method is used to obtain specific "Texton ids" for each pixel. The same process is applied to gray-scale and RGB images of the input to generate "brightness and color maps", respectively.

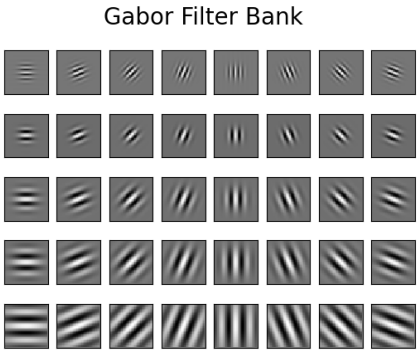


Fig. 3: Gabor Filter Bank

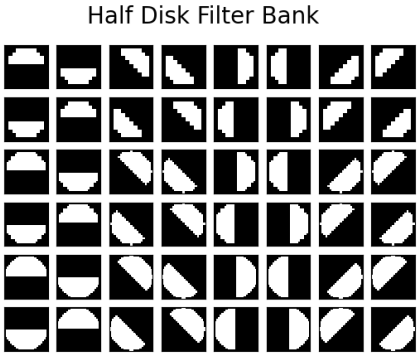


Fig. 4: Half Disk Maks

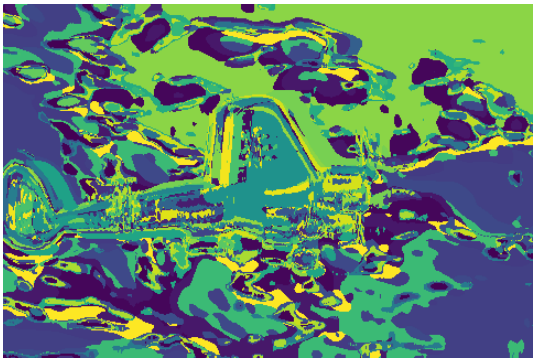


Fig. 5: Texton Map

A brightness map, also known as a grayscale or intensity map, is a single-channel image that represents the brightness or intensity of each pixel in the original image. It is created by converting an RGB image to grayscale, where each pixel's value is determined by the average of its red, green, and blue channels. Brightness maps are often used to detect edges and boundaries in an image, as well as to enhance the visibility of certain features.

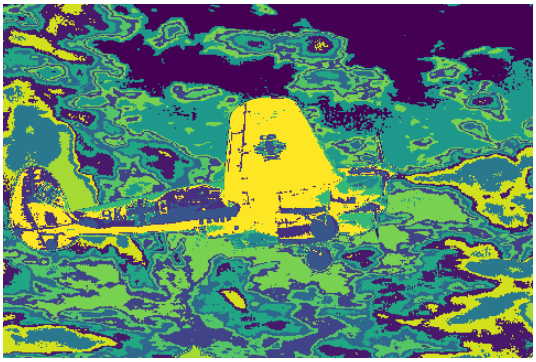


Fig. 6: Color Map

Fig. 7: Brightness Gradient

A color map, on the other hand, is a multi-channel image that represents the color of each pixel in the original image. It can be created by applying a color space transformation to the original image, such as converting an RGB image to the HSV color space. Color maps are often used to segment an image into different regions based on color, as well as to enhance the visibility of certain colors.

Both brightness and color maps are used as a pre-processing step before performing other computer vision tasks such as feature extraction, object detection, and image segmentation.

You can see Texton, Brightness , and Color Maps for Image 1 in Figure 4,5, and 6 respectively

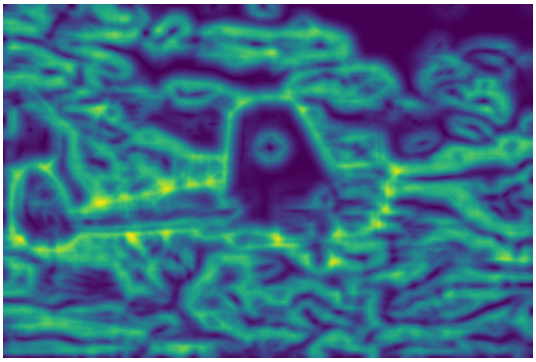


Fig. 8: Color Gradient

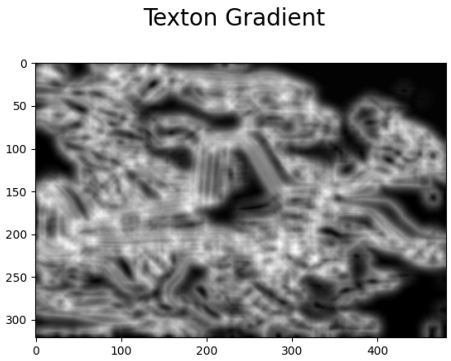


Fig. 9: Texton Gradient

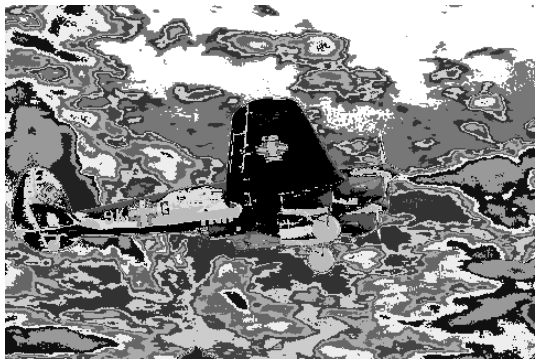


Fig. 10: Brightness Map

C. Gradient map generation

The texton, color, and brightness maps that have been generated are convoluted with pairs of half-disk masks. These half-disk masks are binary images created by rotating a semicircle. We created these masks by first generating a semicircle, then traversing the 2D space of the kernel and calculating the Euclidean distance from the origin for each pixel.

After that these maps were convolved and used with half-disk mask togenerated different gradients in Texton, Color, and Brightness domains. Maps can be seen on figures from 8 to 9.

D. Pb-Lite Output and Results

The gradients calculated in the previous section are utilized, in conjunction with Sobel and Canny techniques, to determine a probabilistic boundary for each image. The gradients from the N-dimensional vector for each gradient image are combined. The resulting 2D maps for our gradients are then combined and used to compute the Hadamard product with weights of 0.5 for each the Sobel and Canny results. Output of the Pb-Lite for the first image along with Sobel and Canny baselines can be seen in Figures from 11 to 13.

The Pb-Lite algorithm allows the user to adjust the amount of texture edges that are visible in the final output by adjusting the contribution of texture gradient, brightness gradient, and color gradient. This makes it more versatile and superior to Sobel and Canny edge detectors, which rely solely on the derivative operator to identify all types of intensity and color changes in images.

II. PHASE 2: Deep Dive on Deep Learning

The base convolutional neural network (CNN) architecture I used performs pretty good.. It consists of several layers, including convolutional layers, activation layers, pooling layers, and fully connected layers. The network starts by using a 2D convolutional layer with 3 input channels, 16 output channels and a kernel size of 3x3, with padding of 1. This layer is used to extract features from the image. The output is then passed through a ReLU activation function, which helps to introduce non-linearity to the model. Then there is another 2D convolutional layer with 16 input channels, 32 output channels and kernel size of 3x3 with padding of 1. This layer is also followed by a ReLU activation function. Then there is a 2D max pooling layer with kernel size of 2x2 which is used for down-sampling the spatial dimensions of the image by taking the maximum value of a 2x2 window. After that, there are two more convolutional layers with similar architecture but with increasing number of filters. Then again a 2D max pooling layer is used for down-sampling. Then the output is passed through a Flatten layer which reshape the output from the previous layers into a 1-D array. Next, there are two linear layers (also called fully connected layers) where the first one has 88128 input units and 100 output units and the second one has 100 input units and 10 output units. Each linear layer is followed by a ReLU activation function. The final output is the prediction of 10 classes. The visualization of the architecture can be seen in Figure 41 and its Accuracy and Loss on Figures 42 and 43. as I said it performed really well with 87 percent accuracy after training for only 10 epochs. You can see the accuracy and confusion matrix in Figure 44.

A. Updated Base Model

In order to further increase accuracy of my model, I tried using more layers, more deeper layers and drop-out as well. Increase the number of filters in the convolutional layers: Increasing the number of filters in the convolutional layers can help to extract more features from the input image, leading to better performance. Adding more layers to the network can help to extract more complex features from the input image. Adding dropout layers can help to reduce over-fitting by randomly dropping out some neurons during training.

After that I implemented different famous Neural Networks Like ResNet,ResNext, and DenseNet but due to lack of time and hardware limitations I was not able to test them.

(a) Sobel 1 (b) Canny 1 (c) Pb-Lite 1

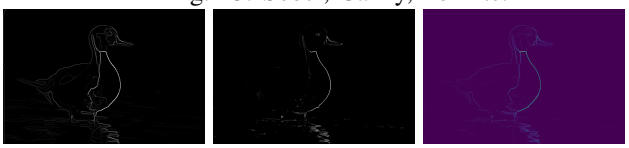
Fig. 11: Sobel, Canny, Pb-Lite.

(a) Sobel 2 (b) Canny 2 (c) Pb-Lite 2

Fig. 12: Sobel, Canny, Pb-Lite.

(a) Sobel 3 (b) Canny 3 (c) Pb-Lite 3

Fig. 13: Sobel, Canny, Pb-Lite.



(a) Sobel 4 (b) Canny 4 (c) Pb-Lite 4

Fig. 14: Sobel, Canny, Pb-Lite.

(a) Sobel 5 (b) Canny 5 (c) Pb-Lite 5

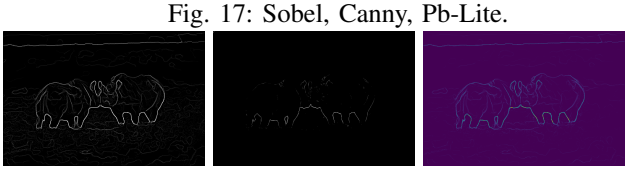
Fig. 15: Sobel, Canny, Pb-Lite.

(a) Sobel 6 (b) Canny 6 (c) Pb-Lite 6

Fig. 16: Two images side by side.

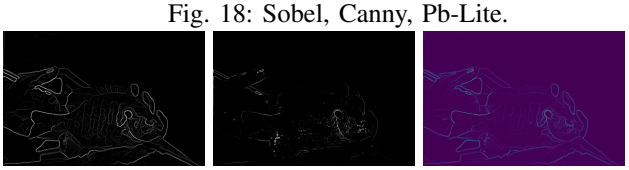
(a) Sobel 7 (b) Canny 7 (c) Pb-Lite 7

Fig. 17: Sobel, Canny, Pb-Lite.

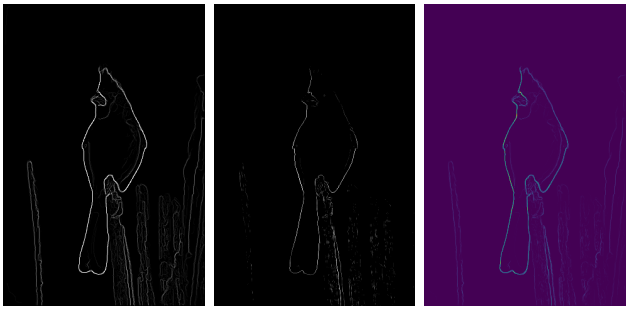


(a) Sobel 8 (b) Canny 8 (c) Pb-Lite 8

Fig. 18: Sobel, Canny, Pb-Lite.

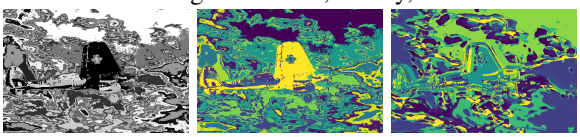


(a) Sobel 9 (b) Canny 9 (c) Pb-Lite 9



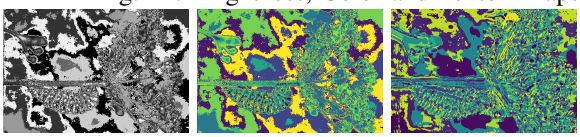
(a) Sobel 10 (b) Canny 10 (c) Pb-Lite 10

Fig. 20: Sobel, Canny, Pb-Lite.



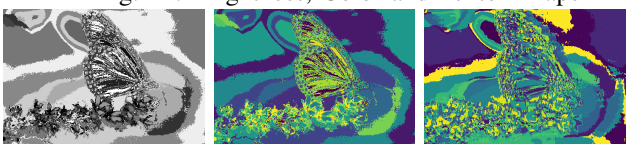
(a) Brightness (b) Color (c) Texton

Fig. 21: Brightness, Color and Texton Maps



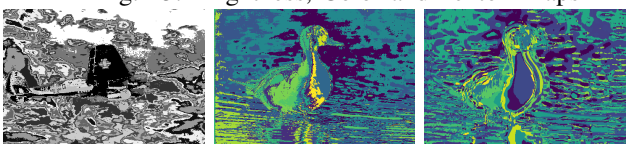
(a) Brightness 2 (b) Color 2 (c) Texton 2

Fig. 22: Brightness, Color and Texton Maps



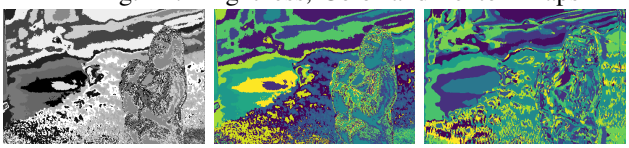
(a) Brightness 3 (b) Color 3 (c) Texton 3

Fig. 23: Brightness, Color and Texton Maps



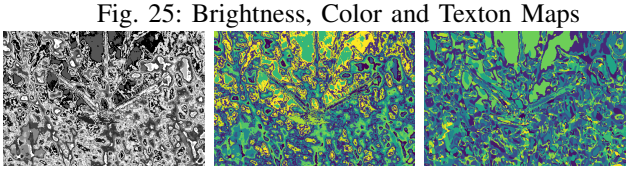
(a) Brightness 4 (b) Color 4 (c) Texton 4

Fig. 24: Brightness, Color and Texton Maps



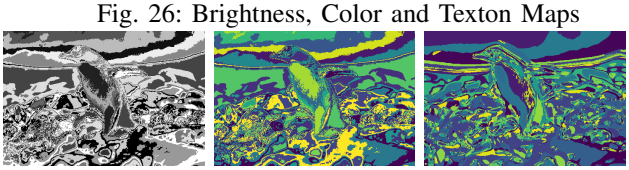
(a) Brightness 5 (b) Color 5 (c) Texton 5

Fig. 25: Brightness, Color and Texton Maps



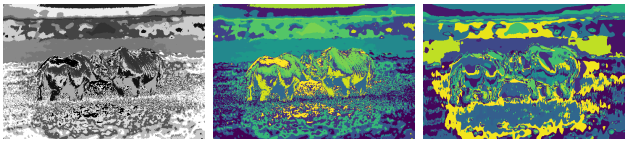
(a) Brightness 6 (b) Color 6 (c) Texton 6

Fig. 26: Brightness, Color and Texton Maps



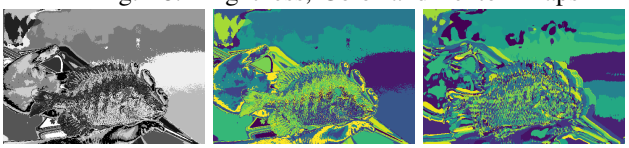
(a) Brightness 7 (b) Color 7 (c) Texton 7

Fig. 27: Brightness, Color and Texton Maps



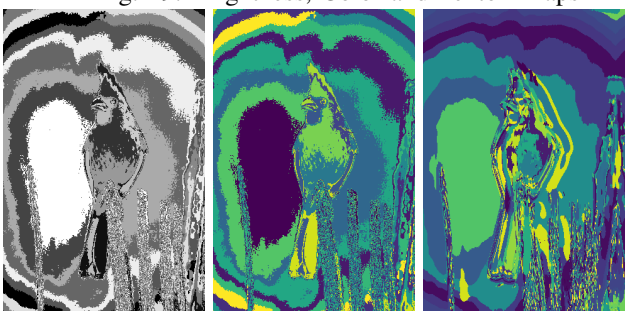
(a) Brightness 8 (b) Color 8 (c) Texton 8

Fig. 28: Brightness, Color and Texton Maps



(a) Brightness 9 (b) Color 9 (c) Texton 9

Fig. 29: Brightness, Color and Texton Maps

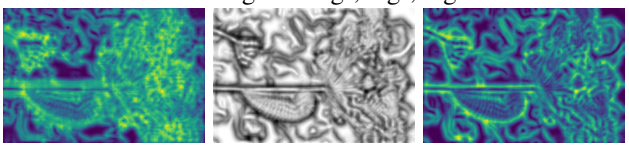


(a) Brightness 10 (b) Color 10 (c) Texton 10

Fig. 30: Brightness, Color and Texton Maps

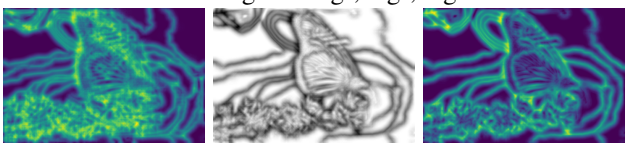
(a) Tg 1 (b) Tb 1 (c) Tc 1

Fig. 31: Tg , Bg , Cg



(a) Tg 2 (b) Tb 2 (c) Tc 2

Fig. 32: Tg , Bg , Cg

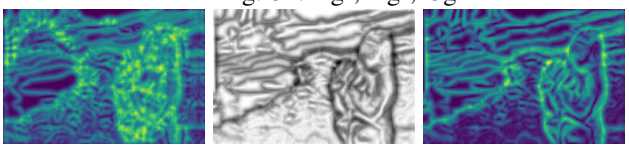


(a) Tg 3 (b) Tb 3 (c) Tc 3

Fig. 33: Tg , Bg , Cg

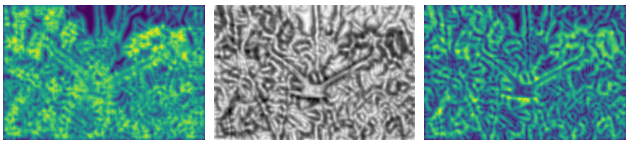
(a) Tg 4 (b) Tb 4 (c) Tc 4

Fig. 34: Tg , Bg , Cg



(a) Tg 5 (b) Tb 5 (c) Tc 5

Fig. 35: Tg , Bg , Cg



(a) Tg 6 (b) Tb 6 (c) Tc 6

Fig. 36: Tg , Bg , Cg

(a) Tg 7 (b) Tb 7 (c) Tc 7

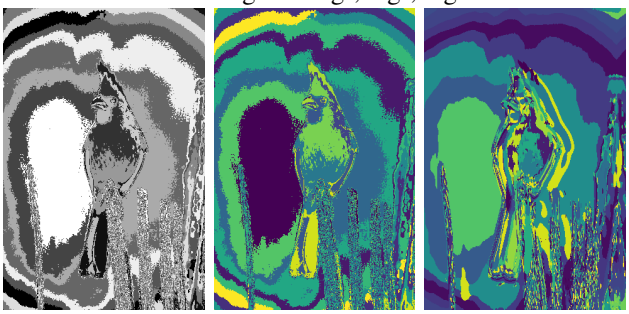
Fig. 37: Tg , Bg , Cg

(a) Tg 8 (b) Tb 8 (c) Tc 8

Fig. 38: Tg , Bg , Cg

(a) Tg 9 (b) Tb 9 (c) Tc 9

Fig. 39: Tg , Bg , Cg



(a) Brightness (b) Color (c) Texton

Fig. 40: Brightness, Color and Texton Maps

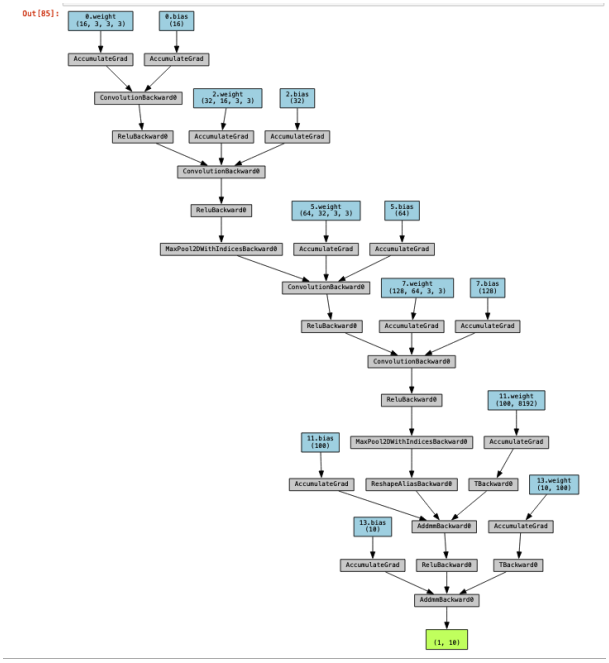


Fig. 41: Base Model Architecture

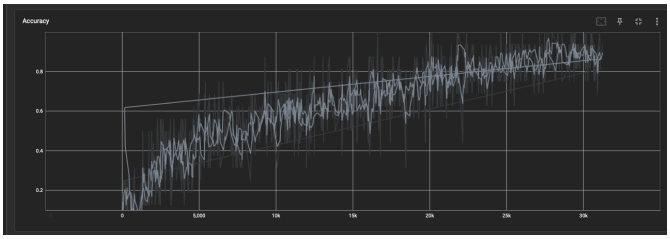


Fig. 42: Base Model Acc

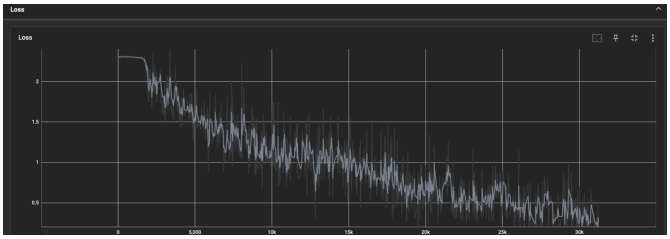


Fig. 43: Base Model Loss

```
Number of parameters in this model are 12

100% 50000/50000 [00:39<0

[3963  100  287   60   74   28   20   44  274  150] (0)
[   9 4707   20   24   16    4   14   11   51  144] (1)
[  60    7 4403  116  174   84   61   52   32   11] (2)
[   9   10  242 3930  145  425   89  105   30   15] (3)
[  20    6  213   87 4330   73   57  179   28    7] (4)
[   4    2  211  366  109 4115   40  113   18   22] (5)
[   4   13  209  169  127   93 4339   24   13    9] (6)
[   5    3  100   73  112   72    7 4602   11   15] (7)
[  37   65   32   21   13    9    7    4 4782   30] (8)
[  12  189   42   41   18   16   19   27   78 4558] (9)
  (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
Accuracy: 87.458 %
```

Fig. 44: Base Model Confusion Matrix