Homework 0: Alohomora

Miheer Diwan

MS Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, 01609
Email: msdiwan@wpi.edu

***Abstract*—This report presents my results for Phase 1 and Phase 2 of Homework 0 for the course CS/RBE 549: Computer Vision and provides a brief analysis of the techniques used. Phase 1 of this homework involves the implementation of a probability of boundary detection algorithm called 'pb-lite' while Phase 2 deals with developing multiple CNNs for a classification task on the CIFAR-10 data set.**

I. Phase 1: Shake My Boundary

In Phase 1 of this homework, I implemented a boundary detector algorithm called 'pb-lite' which is a simplified version of the probability of boundary detection algorithm presented in [1]. The output of the pb-lite algorithm is a per-pixel probability of boundary. Fig. 1 gives an overview of the algorithm. First, we filter the input images with the filter bank and apply k-means clustering to develop the texture, brightness and color maps for an input image. We then compute the texture gradient ($\mathcal{T}_g$), brightness gradient (B_g) and color gradient (C_g). We then calculate the Chi-square distances with the help of half-disk filters. Finally, we combine the information from the features with baseline methods such as Sobel and Canny to get the pb-lite output.

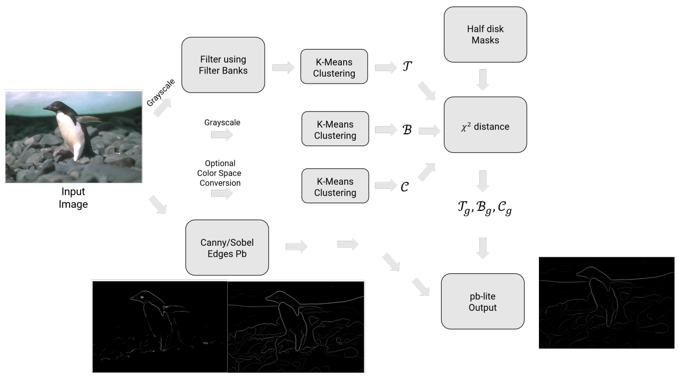


Fig. 1. Overview of pb-lite algorithm

A. Filter Bank

A filter bank consisting of various filters with different scales and orientations was generated for the filtering operation was used to measure the texture properties and to aggregate regional texture and brightness distributions. It consisted of three different sets of filters- Oriented Derivative of Gaussian (DoG) Filters, Leung-Malik Filters and Gabor Filters.

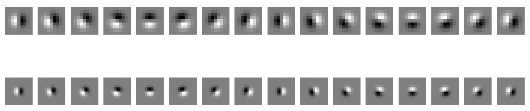


Fig. 2. Oriented DoG Filter Bank

1) Oriented Derivative of Gaussian (DoG) Filters: This filter bank consists of derivative of Gaussian kernels at two scales (σ =1,2), 16 orientations and a kernel size of 9 and 15 respectively. The Gaussian kernel was generated using the following formula:

$$g = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

The Gaussian kernel was then convolved with Sobel kernels to get the derivative in the x and y direction.

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

2) Leung-Malik Filters: This filter bank consists of two filter banks- LM Small (σ =1,$\sqrt{2}$,2,2$\sqrt{2}$) and LM Large (σ =$\sqrt{2}$,2,2$\sqrt{2}$,4) with 48 filters each. Each filter bank consists of 18 first and second DoG filters respectively with an elongation factor of 3, at 6 orientations and 3 scales, 8 Laplacian of Gaussian filters and 4 Gaussian Filters.

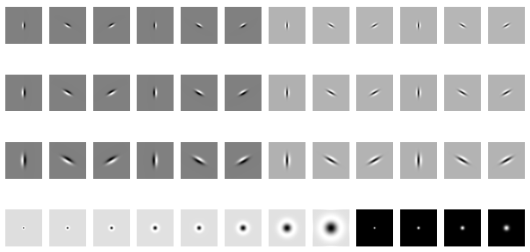


Fig. 3. Leung-Malik Filter Bank

The Laplacian of Gaussian kernel was generated it by convolving the Gaussian kernel with a Laplacian kernel.

$$L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

3) Gabor Filters: Gabor filters are generated by modulating a sinusoidal plane wave with a Gaussian kernel. Multiple Gabor filters were generated using the formula and varying parameters such as kernel size, σ and λ:

$$g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = \exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}).\exp(i(2\pi\frac{x'}{\lambda} + \psi))$$

$$x' = x\cos\theta + y\sin\theta$$
$$y' = -x\sin\theta + y\cos\theta$$

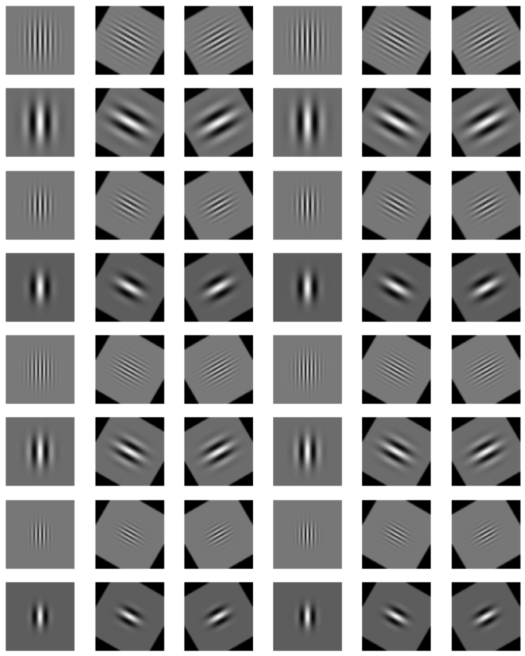


Fig. 4. Gabor Filters of different orientations and parameters

B. Texton Maps (τ), Brightness Maps (B) and Color Maps (C)

Texture Maps were produced by filtering the input images with the filter bank. Each filter produced a filtered output for each image. Essentially, if there were N filters in the filter bank, there were N filter responses at each pixel of the image. In my case, the total filters = 176. We then applied k-means clustering with a bin size = 64 on the filtered responses to get the Texton map.

We used a similar approach to produce the Brightness maps, however, we passed the grayscale images and the bin size = 16. For generating the Color maps, we passed the input image in different color spaces such as RGB or HSV and clustered them with a bin size = 16.

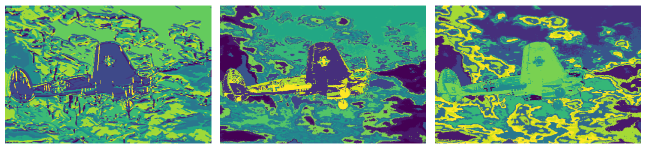


Fig. 5. Texton, Brightness and Colour Maps for Image 1

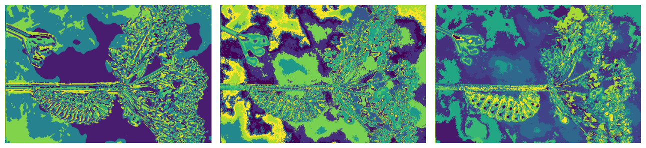


Fig. 6. Texton, Brightness and Colour Maps for Image 2

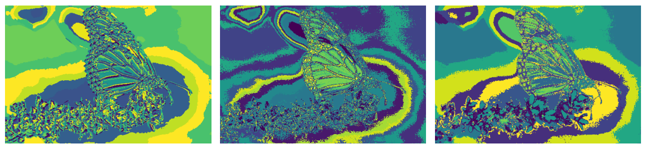


Fig. 7. Texton, Brightness and Colour Maps for Image 3

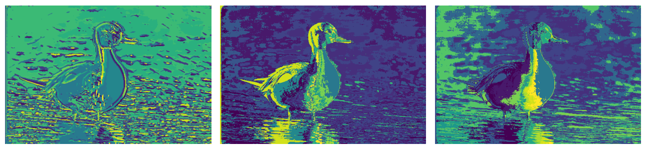


Fig. 8. Texton, Brightness and Colour Maps for Image 4

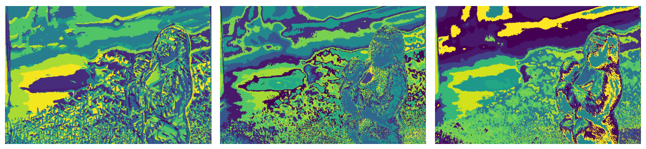


Fig. 9. Texton, Brightness and Colour Maps for Image 5

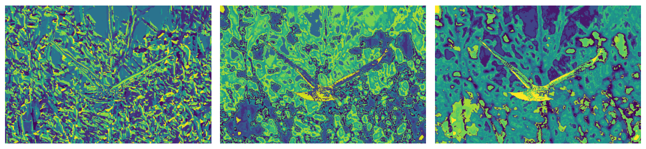


Fig. 10. Texton, Brightness and Colour Maps for Image 6

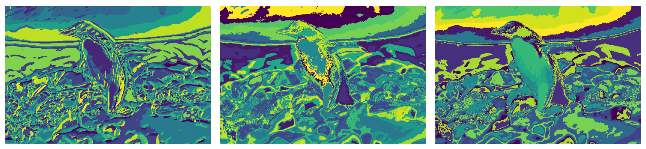


Fig. 11. Texton, Brightness and Colour Maps for Image 7

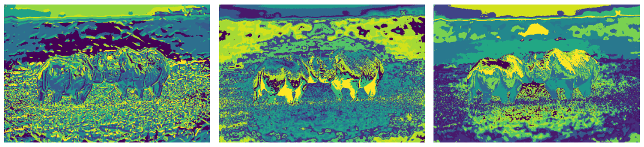


Fig. 12. Texton, Brightness and Colour Maps for Image 8

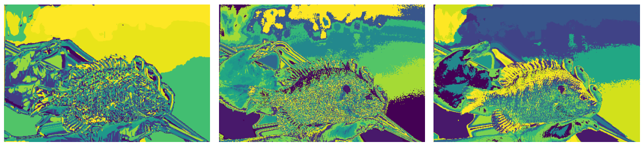


Fig. 13. Texton, Brightness and Colour Maps for Image 9

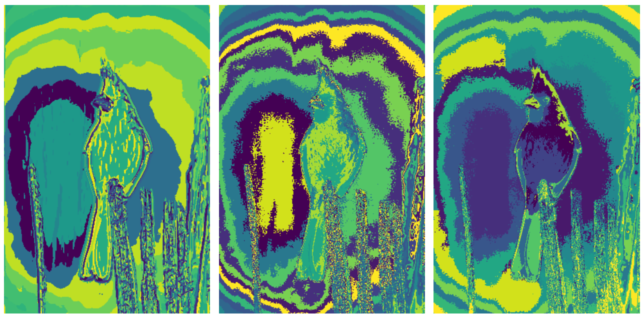


Fig. 14. Texton, Brightness and Colour Maps for Image 10

C. *Gradients - τ_g , Bg and Cg*

The texture gradient (τ_g), Brightness gradient (B_g) and Color gradient (C_g) encode how much the texture, brightness and color distributions are changing at a pixel. We compute these by calculating the χ^2 (Chi^2) distance with the help of half-disk filters. χ^2 distances are calculated with the following formula:

$$\chi^2(g,h) = \frac{1}{2} * \sum_{i=1}^{K} \frac{(g_i-h_i)^2}{g_i+h_i}$$

D. *pb-lite outputs*

Finally, I calculated the pb-lite outputs with the Canny and Sobel baselines by using the following formula:

$$\text{pb} = \frac{\tau_g+B_g+C_g}{3} \odot (w1 * Canny + w2 * Sobel)$$

E. *pb-Lite Output Analysis*

It can be observed that the probability of boundary edges detected with the pb-lite algorithm are better than Sobel baselines in terms of edges and better than Canny baselines in terms of noise reduction. While the edges are not as prominent as the Canny Baselines, there is significant reduction in the background noise and the main object in the image has been identified successfully. To improve the detected edges, I tried experimenting with the number and orientations of Gabor filters. I found that more orientations gave a better result. However, using too many Gabor filters also increased the background edges a bit. It should also be noted that the clustering step randomly selects points and creates clusters in every run of the program, thus slightly varying the results.

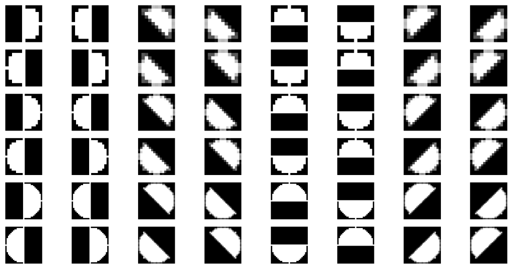


Fig. 15. Half-Disk Filters

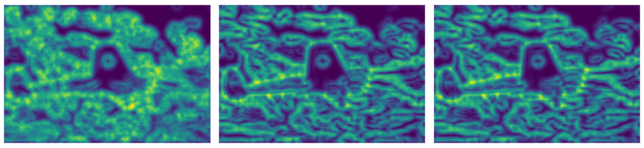


Fig. 16. τ_g, Bg and Cg for Image 1

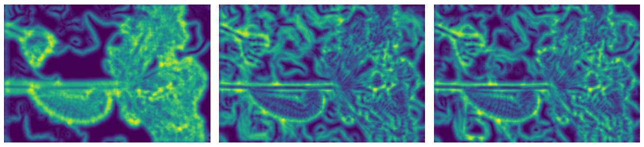


Fig. 17. τ_g, Bg and Cg for Image 2

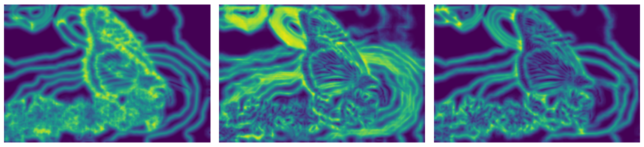


Fig. 18. τ_g, Bg and Cg for Image 3

II. Phase 2: Deep Dive on Deep Learning

In this phase, we implemented multiple CNNs to perform classification task on the CIFAR-10 dataset. The dataset had 10 classes, 50000 training images and 10000 test images.

A. *Base Model*

1) Network Architecture: My base model consists of three 2-D convolutional layers with a kernel size of 3 and three linear layers activated by a ReLU activation function and three MaxPooling layers. The input images are RGB images of size 3*32*32. The first conv2D layer takes 3 as input and outputs 32 channels. The next layer has 32 input and 16 output channels. The last conv2D layers has 16 input

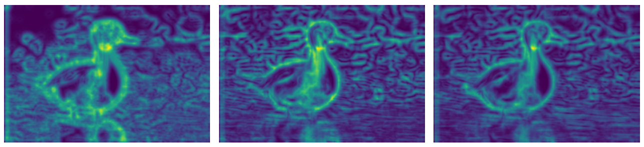


Fig. 19. τ_g, Bg and Cg for Image 4

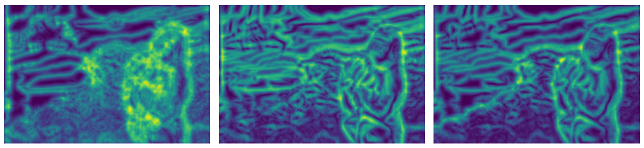


Fig. 20. τ_g, Bg and Cg for Image 5

and 8 output channels. The model also contains 3 fully connected layers. The last linear layer has 10 output channels because we have 10 classes. It is followed by a Softmax layer which will normalize the outputs. The loss function used was Cross entropy Loss and the optimizer used was AdamW. The netwrok was trained for 20 epochs. The accuracy of this network was 44.16 %.

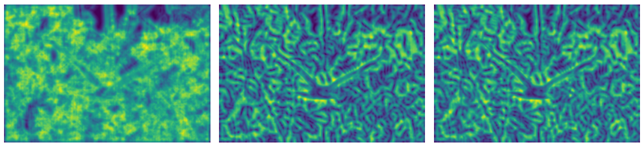


Fig. 21. τ_g, Bg and Cg for Image 6

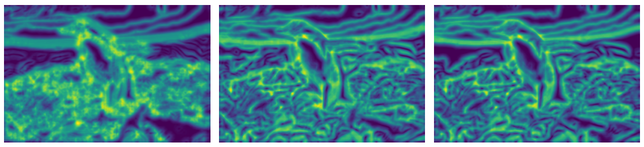


Fig. 22. τ_g, Bg and Cg for Image 7

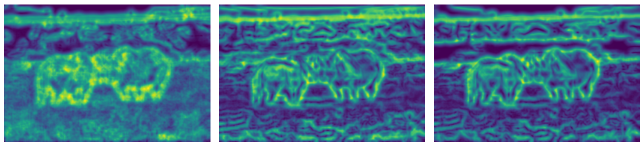


Fig. 23. τ_g, Bg and Cg for Image 8

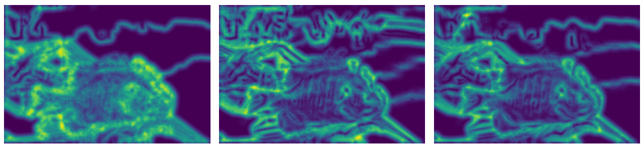


Fig. 24. τ_g, Bg and Cg for Image 9

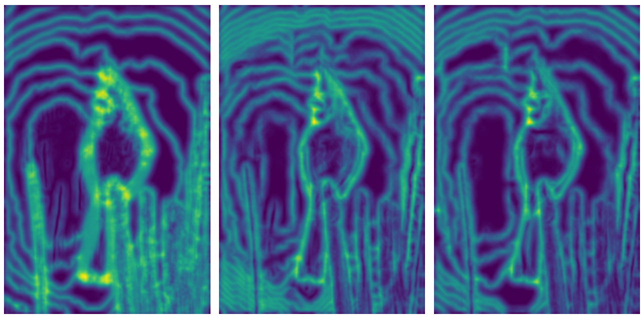


Fig. 25. τ_g, Bg and Cg for Image 10

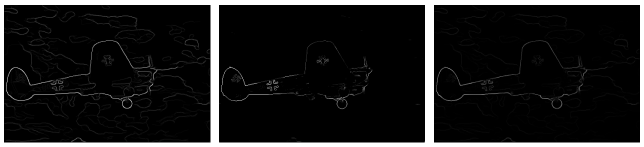


Fig. 26. Canny, Sobel and pb-lite output for Image 1

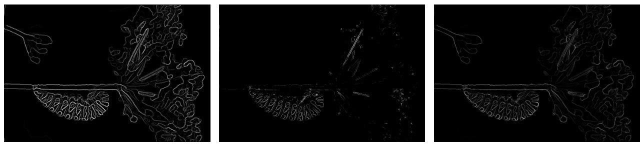


Fig. 27. Canny, Sobel and pb-lite output for Image 2

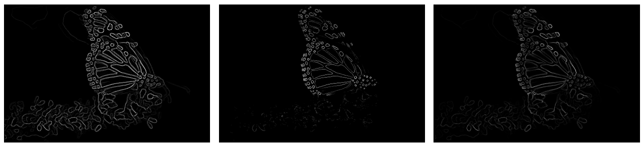


Fig. 28. Canny, Sobel and pb-lite output for Image 3

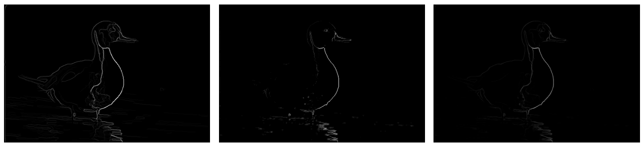


Fig. 29. Canny, Sobel and pb-lite output for Image 4

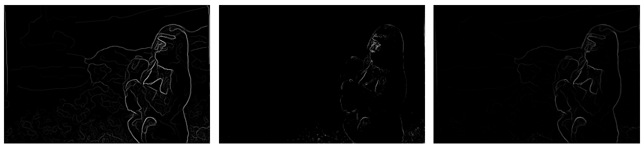


Fig. 30. Canny, Sobel and pb-lite output for Image 5

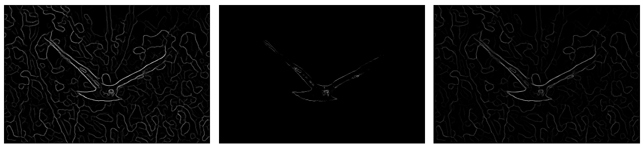


Fig. 31. Canny, Sobel and pb-lite output for Image 6

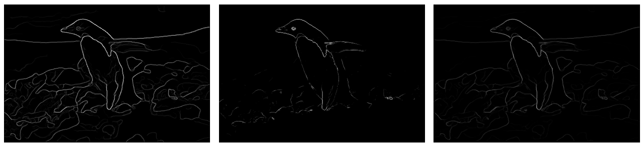


Fig. 32. Canny, Sobel and pb-lite output for Image 7

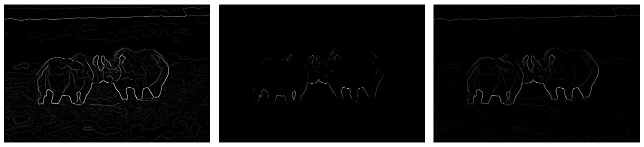


Fig. 33. Canny, Sobel and pb-lite output for Image 8

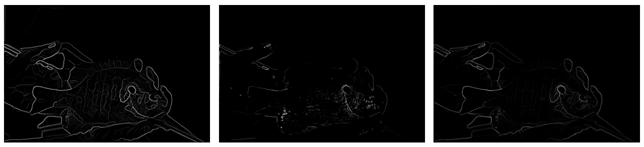


Fig. 34. Canny, Sobel and pb-lite output for Image 9

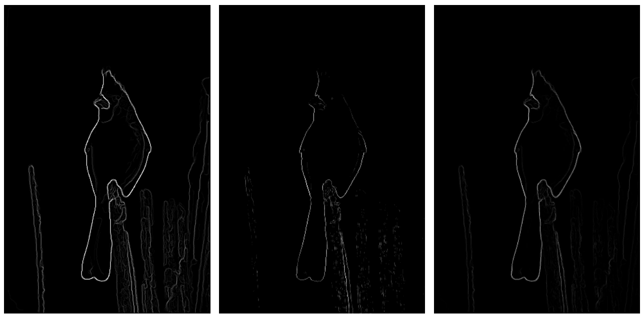


Fig. 35. Canny, Sobel and pb-lite output for Image 10

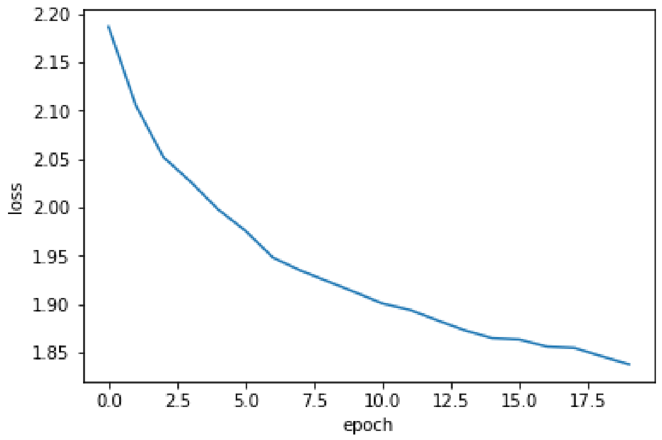
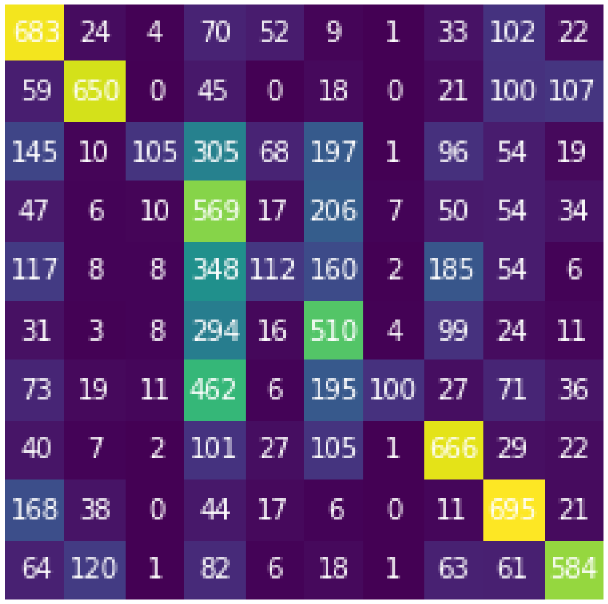


Fig. 36. Loss Vs Epochs



683	24	4	70	52	9	1	33	102	22
59	650	0	45	0	18	0	21	100	107
145	10	105	305	68	197	1	96	54	19
47	6	10	569	17	206	7	50	54	34
117	8	8	348	112	160	2	185	54	6
31	3	8	294	16	510	4	99	24	11
73	19	11	462	6	195	100	27	71	36
40	7	2	101	27	105	1	666	29	22
168	38	0	44	17	6	0	11	695	21
64	120	1	82	6	18	1	63	61	584

Fig. 37. Confusion Matrix