CMSC 733
Computer Processing of Pictorial Information
Report - Homework 0

Rashmi Kapu
A. James Clark School of Engineering
University of Maryland
College Park, Maryland
Email: rashmik@umd.edu

I. Phase 1

This phase addresses the challenge of boundary detection in computer vision, noting its importance for understanding object transitions within images. Traditional edge detection methods like Canny and Sobel primarily rely on intensity changes, which may lead to false positives in textured areas. To address this limitation, this approach proposes a simplified version of the pb (probability of boundary) algorithm, which considers brightness, color, and texture information across different scales in the image. This approach aims to improve boundary detection accuracy by incorporating additional features beyond intensity discontinuities. Given below are the steps followed to achieve this :

- Generating filter banks
- Generating Texton maps
- Generating Brightness and Color maps
- Gradients using half-disk masks and χ^2 distances
- Combining the gradients with Canny and Sobel baselines to get PB-lite outputs

A. Generating filter banks

Filter banks are created in order to detect certain features in the image. The convolution operation of these filters on the image detect the feature corresponding to the filter in the image (translation invariant) and produce a map with intensity values. Higher the intensity, higher the chances of the feature to be present in the image. We use this property to detect edges in our images.

- Derivative of Gaussian (DoG) filter bank:
  The derivative of Gaussian (DoG) filter bank is a set of filters derived from the Gaussian function by taking it's derivatives with respect to space (x and y coordinates). Since our inputs are discrete and not continuous, derivative translates to convolving a simple Sobel filter with a Gaussian kernel. Two scales (standard deviations) and sixteen orientations were taken. For one value of standard deviation, a Gaussian kernel is created and rotated 16 times by approximately 30^0. This can be seen in Figure1.
- Leung-Malik Filter Bank:
  The Leung-Malik (LM) filter bank comprises 48 filters, including first and second-order derivatives of Gaussians, Laplacian of Gaussian (LOG) filters, and Gaussians. These filters capture multi-scale and multi-orientation information in images. In LM Small (LMS), filters span scales σ = [1,$\sqrt{2}$,2,2$\sqrt{2}$], while in LM Large (LML), scales are σ=[$\sqrt{2}$,2,2$\sqrt{2}$,4]. The utility of the LM filter bank lies in its ability to capture multi-scale and multi-orientation information in images, facilitating edge detection tasks and texture analysis. Moreover, its flexibility in scale and orientation enables the LM filters to adapt to a wide range of image structures and characteristics, enhancing their effectiveness. The first and second-order derivatives are generated using convolutions with the Sobel operator, at the first three scales with an elongation factor of 3, that is $\sigma_x = \sigma$ and $\sigma_y = 3\sigma_x$. Laplacians are obtained by convolving the Gaussian kernels with the Laplacian operator. This can be seen in Figure2.
- Gabor filter bank:
  A Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave. 5 values of σ were given

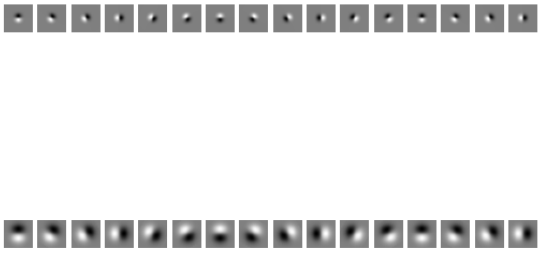


Fig. 1: DoG filters with two scales and sixteen orientations

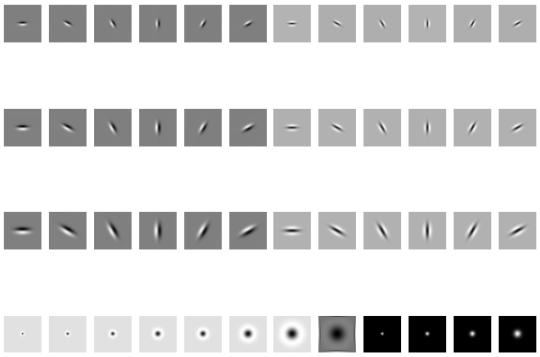


Fig. 2: LM filters, the first 12 images in first 3 rows are the first and second order derivatives at 3 scales with elongation factor. The last row contains 4 Laplacians for σ and 3σ each and 4 Gaussian kernels at the given scales.

with 10 rotations which can be seen in Figure3.

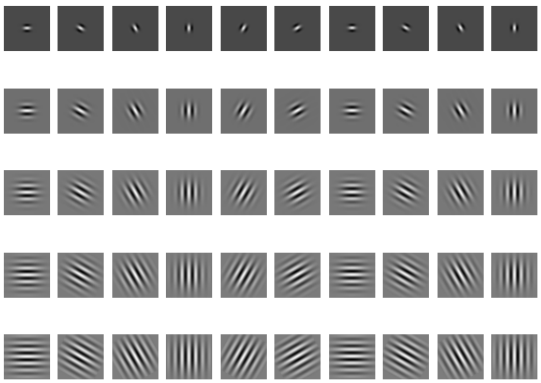


Fig. 3: Gabor filters, for 5 different Standard deviations, 10 orientations each.

B. Generating Texton maps

Texton maps are essential for encoding texture information in images efficiently. They enable feature extraction by representing patterns across various scales. This is done by convolving the filters in the filter bank with the input images. This gives us n matrices for each image if n is the number of filters being used. Since the size of the array obtained will be (no.of images, (size of image), no.of filters), it becomes increasingly complex to work with such lengthy arrays. Hence, we use a unique Texton ID for each pixel of every image to reduce the dimensionality. This is done using K-Means clustering. The pixels are classified into clusters using cluster IDs. The algorithm from Scikit learn's (sklearn.cluster.KMeans) does the job by categorizing these pixels based on the values they have after convolving with every filter. The images are convolved with all the filters in the filter maps and cluster them (100 clusters worked relatively better). Now, similar pixels in the images grouped into one category and hence the same label. We can visualise this to find out the different elements in the image, as shown in Figure 4 and Figure 5.

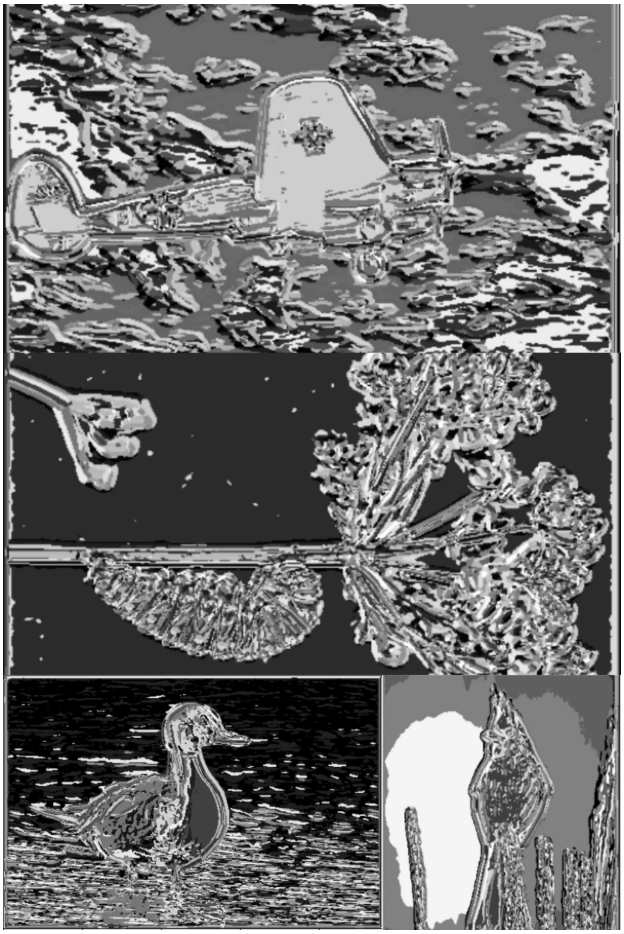


Fig. 4: Texton maps of Images 1,2,10,4 (clockwise)

C. Generating Brightness and Color maps

These maps provide valuable insights into the distribution of pixel intensities and colors within each image, facilitating subsequent analysis and processing tasks.

To generate the brightness maps, first convert the images to grayscale to focus solely on their intensity information. Then, applying k-means clustering on the brightness maps partitions the grayscale intensity values into 16 distinct clusters, representing different levels of brightness across the images. Each pixel in the resulting brightness map is assigned a label corresponding to its cluster centroid, effectively quantizing the intensity distribution. The results for the same are shown in Figure 6 and 7.

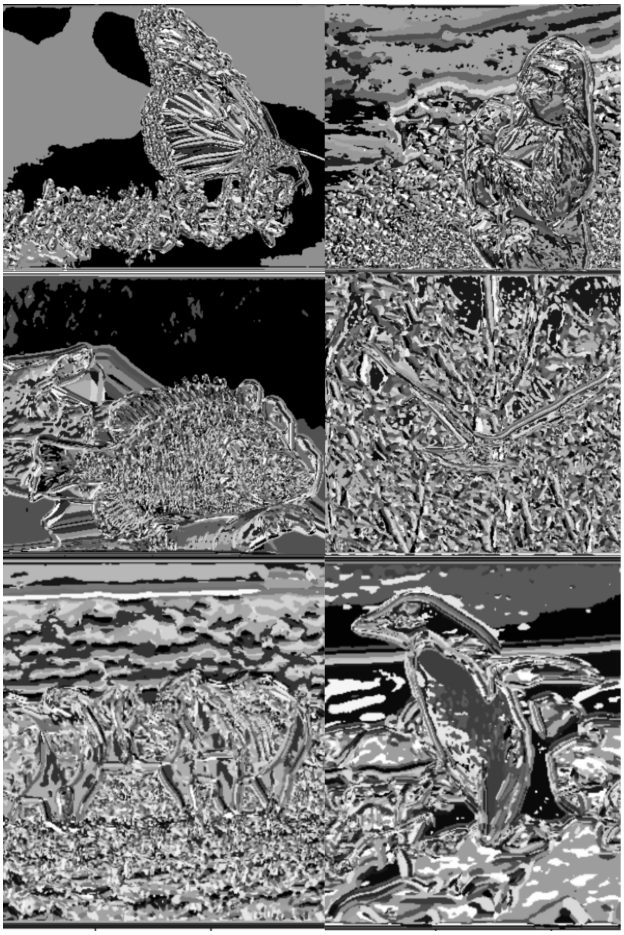


Fig. 5: Texton maps of Images 3,5,6,7,8,9

In generating color maps, each channel (red, green, and blue) of the RGB color space is considered separately. Applying k-means clustering with 16 clusters to each channel independently, we partition the color space into distinct regions based on intensity levels. The resulting color maps provide insights into the distribution of colors within the image, with pixels labeled according to their nearest cluster centroid in each channel. Refer to Figure 8 for a sample output of 4 images.

D. Gradients using half-disk masks and χ^2 distances

In order to compute the χ^2 distances, the initial step is to generate the half-disk masks. These are generated using 4 different kernel sizes and 18 orientations. For each kernel size, radius = half of size-1. The generated half-disk mask is flipped. Then this kernel is rotated, and that is flipped. The rotation occurs 7 times to each of the 4 kernel sizes. Half-disk masks generated are shown in 9. Instead of looping over each pixel neighborhood and aggregating counts for histograms, these half-disks are convolved with the Texton, brightness and color maps to compute χ^2 distances. The variable g is given by left half-disk convolution, and h by right.

E. Combining the gradients with Canny and Sobel baselines to get PB-lite outputs

After computing the above steps to get

$$T_g, B_g, C_g$$

, these can be used to calculate the PB-lite. The provided Canny and Sobel baselines are also used to get the best output.

$$\text{PbEdges} = (T_g + B_g + C_g)^3 \odot (w_1 * \text{cannyPb} + w_2 * \text{sobelPb})$$

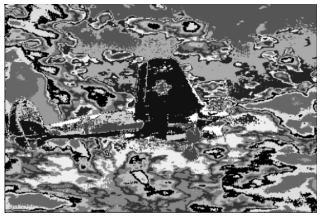
In this equation:

- PbEdges represents the final output.
- T_g, B_g, and C_g are the texton, brightness, and color maps, respectively.
- w_1 and w_2 are the weights for the Canny and Sobel edge maps (taken as 0.5).
- $*$ denotes convolution, and $\odot$ represents element-wise multiplication.

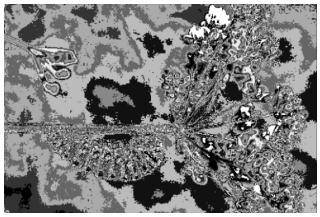
The Sobel and Canny operators excel at detecting edges based on gradient magnitude and intensity discontinuities, respectively. By integrating them with the T, B, and C maps, which capture texture, brightness, and color information, respectively, the model gains a more comprehensive understanding of image structure. This fusion leverages the complementary strengths of each component, allowing for more robust boundary detection across a variety of image types and conditions. This understands textures and does not detect them as edges. Consequently, the resulting PbEdges output provides a richer representation of object boundaries. Figure 10 and 11 show the PBlite output of the given images. In few of these images, it can be seen that edges are lighter (less intensity) at some points, this simply means that the probability of an edge being present at that point is lower than that of brighter ones.

F. How is PBlite better than Sobel and Canny edge detectors?

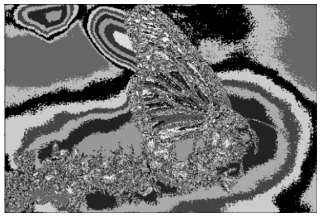
In Figure 12, it can be observed that Canny and Sobel baselines detect noises and textures as edges, while PbLite seems to look robust to noise and distinguishes well between textures and edges. PbLite demonstrates superiority over traditional edge detection methods like Canny and Sobel due to its sophisticated approach. While Canny and Sobel rely primarily on intensity gradients to identify edges, PbLite employs a more comprehensive strategy. By integrating analyses of color, texture, and brightness variations across the image, PbLite achieves more precise boundary detection. This multifaceted approach enables PbLite to discern genuine object boundaries amidst complex visual environments better than its predecessors. Additionally, PbLite's robustness against false positives, such as those caused by shadows or irregular textures, contributes to its superior performance, making it robust to noise. Consequently, PbLite emerges as a better choice for accurate boundary detection in Computer Vision.



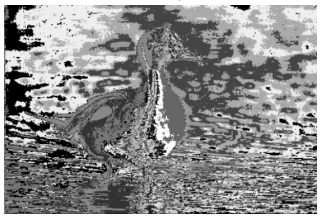
(a) Image 1



(b) Image 2

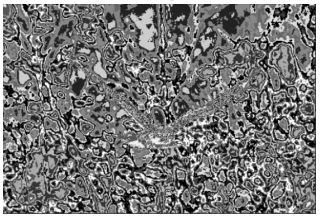


(c) Image 3

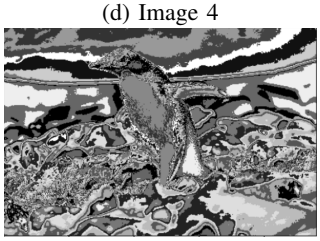


(d) Image 4

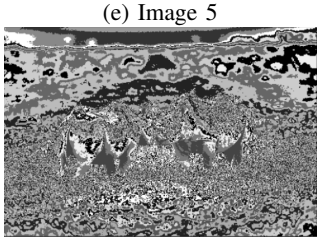
(e) Image 5



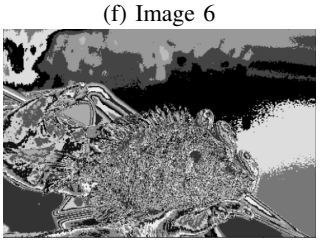
(f) Image 6



(g) Image 7



(h) Image 8



(i) Image 9

Fig. 6: Brightness maps

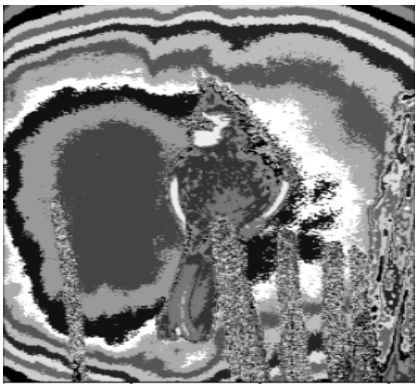


Fig. 7: Brightness map of Image 10

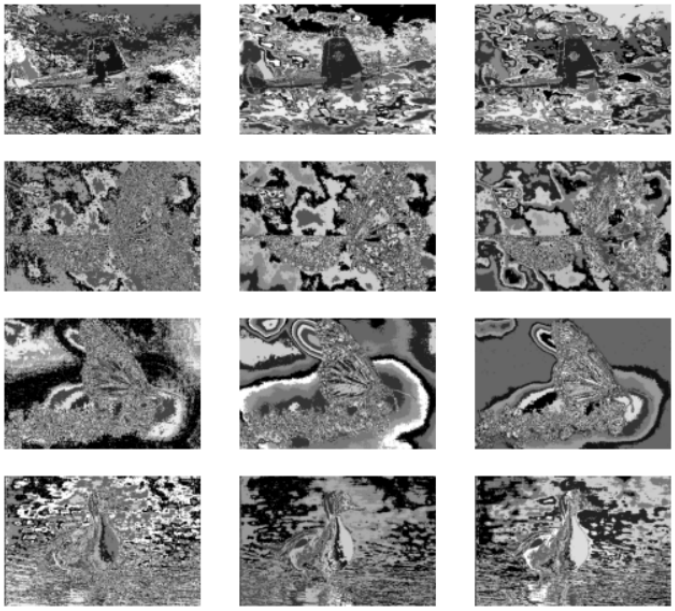


Fig. 8: Red, green and blue channel-wise color maps (stacked column-wise for each image) for input images 1,2,3,4.

II. Phase II

This part focuses on developing multiple neural network architectures and conducting a comparative analysis across various metrics such as the number of parameters, accuracies on both training and test sets. Additionally, providing a comprehensive examination to elucidate why certain architectures outperform others. The training is done on a randomised version (50,000 images) of CIFAR10 image dataset.

- First network is a basic model. The designed two-layer convolutional neural network (CNN) comprises two convolutional layers followed by a fully connected layer with softmax activation for classification. The first convolutional layer uses a kernel size of 5x5, consists of 32 filters, while the second convolutional layer utilizes a kernel size of 3x3 with 16 filters. The purpose of the convolutional layers is to extract meaningful features from the input images. By applying multiple filters to the input image,

Fig. 9: Half-disk masks for kernel sizes = [7,11,25,35] for 7 orientations

these layers can detect patterns and spatial hierarchies in the data. The use of multiple filters allows the network to learn diverse feature representations at different levels of abstraction. Following each convolutional layer, a Rectified Linear Unit (ReLU) activation function is applied to introduce non-linearity into the network. This helps the network learn complex relationships between features in the data. After the convolutional layers, a fully connected layer is employed to perform classification based on the extracted features. The softmax activation function is used in the output layer to produce probability distributions over the classes, enabling the network to output the likelihood of each class. The structure of this network can be seen in Figure13

- ResNet-18 is a convolutional neural network (CNN) architecture proposed by Microsoft Research for image classification tasks. It is part of the ResNet (Residual Network) family of architectures, known for its deep structure and skip connections, which alleviate the vanishing gradient problem during training. The ResNet-18 architecture consists of 18 layers, including convolutional layers, pooling layers, batch normalization layers, ReLU activation functions, and a fully connected layer for classification. It is composed of several residual blocks, each containing two or three convolutional layers. The main innovation of ResNet-18 lies in its use of residual blocks, which allow for the training of very deep networks by mitigating the degradation problem. In a residual block, the input to a layer is added to its output through a skip connection, enabling the network to learn residual functions rather than directly learning the desired mappings. Specifically, ResNet-18 comprises four residual blocks, each containing multiple convolutional layers with a stride of 1 and a kernel size of 3x3. The number of filters in each convolutional layer gradually increases from 64 to 512, allowing the network to capture increasingly complex features as information flows through the layers. Additionally, max pooling layers are employed to downsample the feature maps and reduce spatial dimensions, helping to increase the receptive field of the network while decreasing computational complexity. Batch normalization layers are used to improve the training stability and convergence speed of the network by normalizing the activations of each layer.
  At the end of the network, a global average pooling layer aggregates the feature maps into a one-dimensional vector, which is then fed into a fully connected layer with softmax activation for classification. Figure14 shows the architecture.

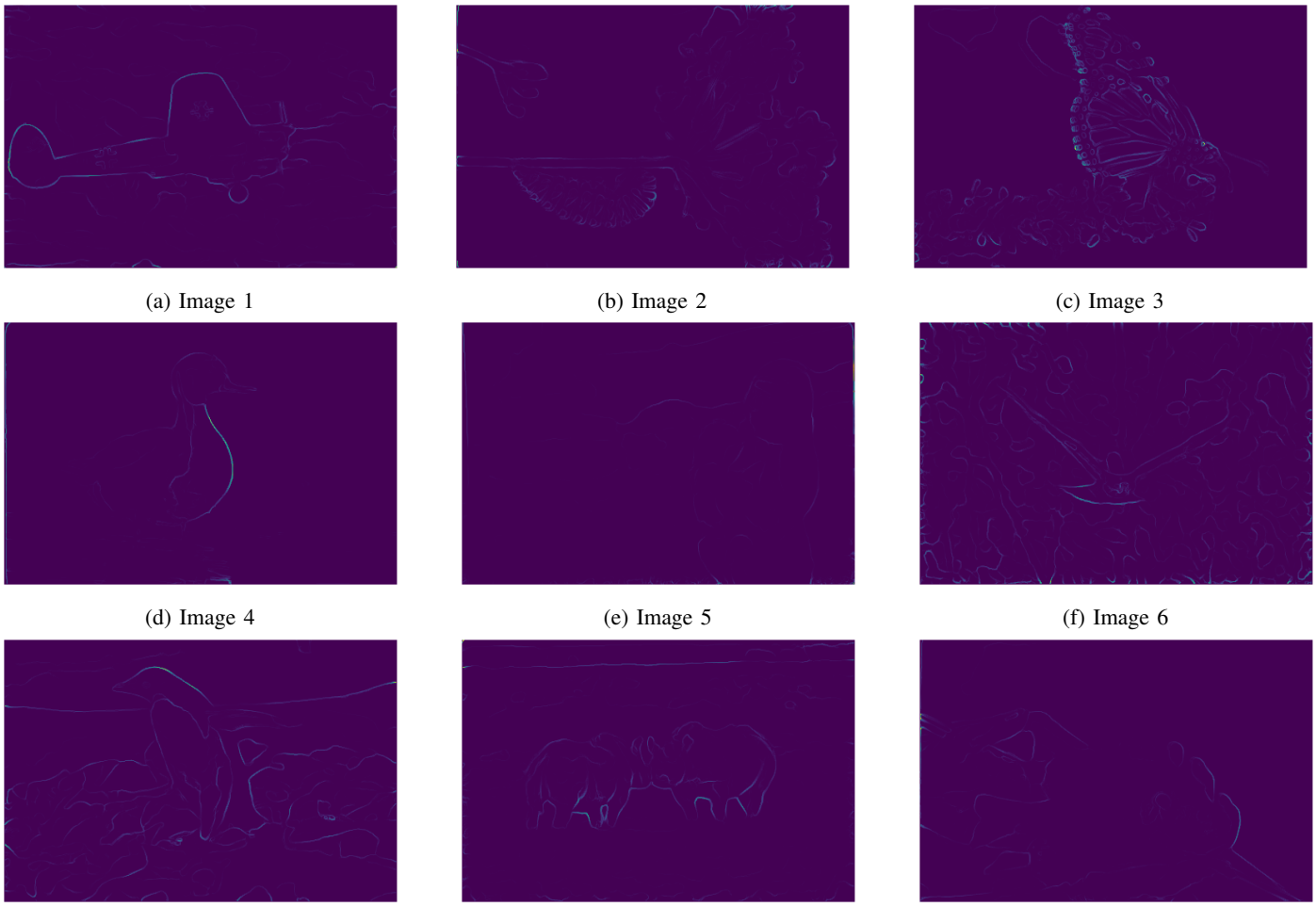
A. Analysis of the networks

Below are the summaries for the models trained :

- **Basic Model: (2 layer Conv net)**
  Framework: TensorFlow 2.x with compatibility functions for TensorFlow 1.x
  Training epochs: 5
  Optimizer : Adam
  Batch size : 8
  Test accuracy: Roughly 10%
  Metrics: Mean Squared Error
  Learning rate: 1×10^{-3}
  Remarks: The basic model achieved low test and train accuracy, indicating potential underfitting. Figure 15 shows the plots of Accuracy and loss over epochs. Figure 17 shows the confusion matrix obtained upon inferencing this trained model on test images.
- **ResNet Model:**
  Test accuracy: Roughly 50%
  Train accuracy: Greater than 50%
  Optimizer : Adam
  Batch size : 8
  Training epochs: 5
  Learning rate: 1×10^{-3}
  Remarks: The ResNet [1] model achieved a test accuracy of roughly 50%, indicating better performance compared to the basic model. Additionally, the train accuracy is also greater than 50%, suggesting that the model is not overfitting. The number of epochs can be increased in order to get a better model. 5 epochs were set due to limited computational units. Figure 16 shows the plots of Accuracy and loss over epochs. Figure 18 shows the confusion matrix obtained upon inferencing this trained model on test images.

III. Conclusion

We saw in Phase I that by combining the Sobel and Canny operators with the T, B, and C maps, which capture texture, brightness, and color information, respectively, the model enhances its ability to detect edges based on gradient magnitude and intensity discontinuities. This integration harnesses the complementary strengths of each component, leading to more



(a) Image 1 (b) Image 2 (c) Image 3

(d) Image 4 (e) Image 5 (f) Image 6

(g) Image 7 (h) Image 8 (i) Image 9

Fig. 10: Final outputs : PBlite

robust boundary detection across diverse image types and conditions. By leveraging this fusion approach, the model gains a deeper understanding of image structure while avoiding the misinterpretation of textures as edges. As a result, the PbEdges output provides a more nuanced representation of object boundaries. In some images, edges may appear lighter, indicating a lower probability of an edge being present compared to brighter areas.

In Phase II, it is observed that ResNet has better accuracy over the basic 2 layer Convolutional network. This is because of a deeper architecture [2] and more layers of batch normalisation, pooling and non-linearities which makes the model more expressive as well as generalize better on unseen data. However, due to computational limitations and compatibility issues with TensorFlow 1, model used only 5 epochs to train. This can be further imrproved by increasing the number of epochs and also building other State-of-the-Art models other than ResNet18, which will be the further work of this paper.

REFERENCES

[1] He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. (2015)

[2] https://medium.com/analytics-vidhya/resnet-understand-and-implement-from-scratch-d0eb9725e0db

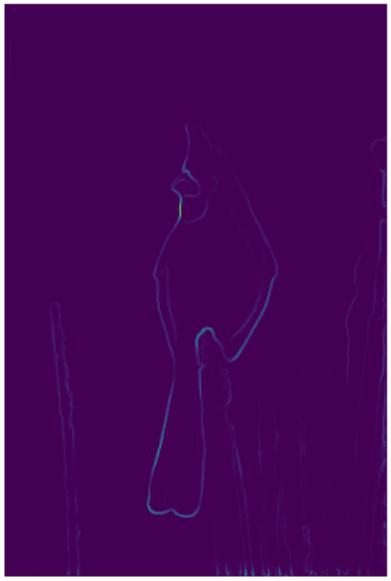


Fig. 11: PBlite output for Image 10

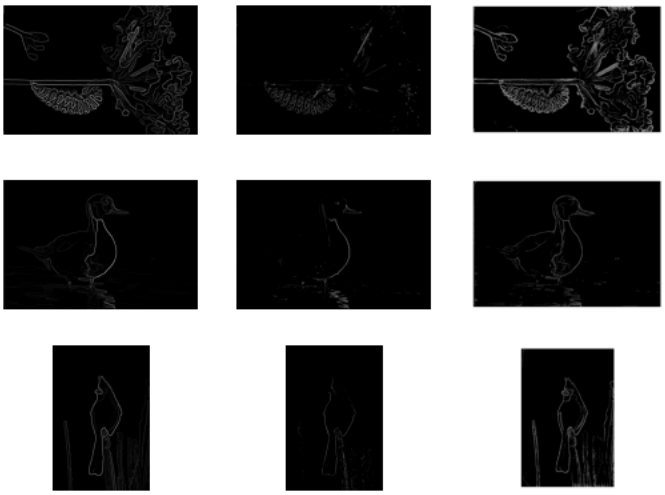


Fig. 12: Comparisons among Canny, Sobel and PBLite outputs in the order (Images 2,4,10). PBlite outputs are enhanced by assigning white color values to all pixels whose probabilities from PBLite algorithm are greater than a threshold.

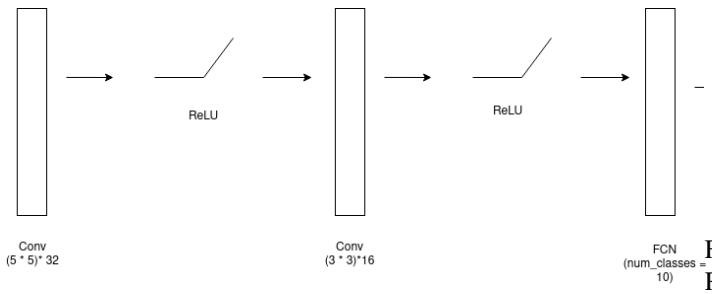


Fig. 13: Basic model (2 layer Convolution network) structure

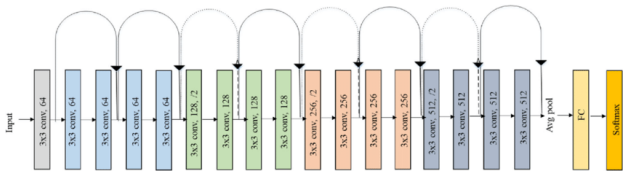


Fig. 14: Model 2 (RESNET18) structure

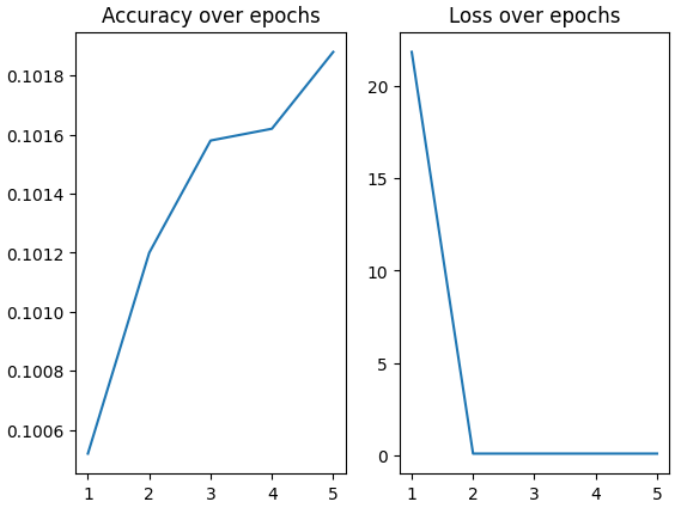


Fig. 15: Accuracy and Loss plots over epochs : Model 1, two layer Conv net

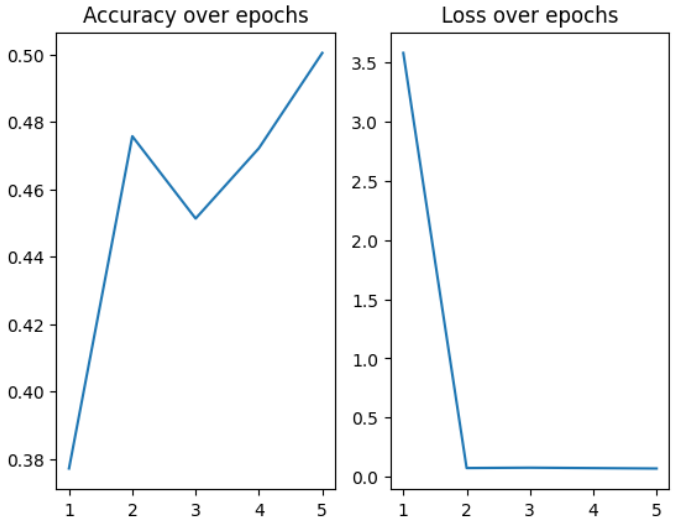


Fig. 16: Accuracy and Loss plots over epochs : Model 2, Resnet18

Class	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	1000	0
1	0	0	0	1	0	0	0	0	999	0
2	0	0	0	0	0	0	0	0	999	1
3	0	0	0	0	0	0	0	0	997	2
4	0	0	0	0	0	0	0	0	1000	0
5	0	0	0	0	0	0	1	999	0	0
6	0	0	0	0	0	0	0	0	1000	0
7	0	0	0	0	0	1	0	0	999	0
8	0	0	0	0	0	0	0	0	1000	0
9	0	0	0	0	0	0	0	0	1000	0

Fig. 17: Confusion matrix obtained for model 1

Class	0	1	2	3	4	5	6	7	8	9
0	476	23	39	71	74	7	40	36	92	142
1	9	492	4	23	10	7	38	22	21	374
2	53	16	75	214	319	63	128	70	11	51
3	7	12	36	349	96	164	176	73	5	82
4	24	2	33	133	470	13	167	126	6	26
5	1	6	19	312	85	328	114	97	7	31
6	1	11	21	100	142	14	639	29	4	39
7	4	0	9	104	80	68	38	650	1	46
8	105	64	11	52	29	10	28	23	528	150
9	18	51	5	38	17	10	29	42	28	762

Fig. 18: Confusion matrix obtained for model 2