RBE549 Computer Vision HW0

Rajus Nagwekar
RBE Department
Worcester Polytechnic Institute
Worcester, MA
Email: rmnagwekar@wpi.edu
Using 1 late day

***Abstract*—In this assignment, we implement boundary detection using the probability of boundary (pb-lite) algorithm. For phase 2, we implement a few neural network architectures for classification problems to compare the models with each other.**

I. Phase 1

We implement the probability of boundary (pb-lite) algorithm. The pb-lite algorithm considers texture and color discontinuities which enables it to out-perform traditional methods like Sobel and Canny which only look for intensity discontinuities. We implement the probability of boundary (pb-lite) algorithm. The pb-lite algorithm considers texture and color discontinuities which enables it to out-perform traditional methods like Sobel and Canny which only look for intensity discontinuities.

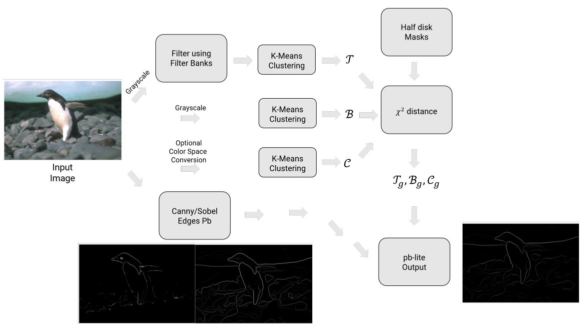

Fig. 1: Pb-lite Pipeline

mds
August 26, 2015

A. Filter Banks

The initial stage of the pb lite boundary detection process involves using a collection of filter banks on the image. Three distinct filter bank sets will be established for this purpose. After filtering the image with these filters, a texton map, which illustrates the texture within the image, will be generated by grouping together the responses from the filters.

1) Oriented DoG Filters: The Derivative of Gaussian filters are created by convolving a Sobel filter and a Gaussian kernel to take different orientations of the output. The filter bank contains DoG filters of size 15x15 at 16 orientation and 2 different scales.

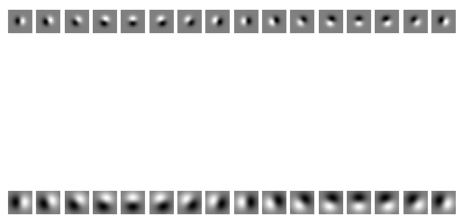

Fig. 2: DoG Filters

2) Leung-Malik Filters: The Leung-Malik filters, also known as LM filters, are a collection of 48 filters that vary in scale and orientation. It includes 36 filters that are first and second order derivatives of Gaussians at 6 orientations and 3 scales, 8 filters that are Laplacian of Gaussian (LOG) and 4 filters that are Gaussians.

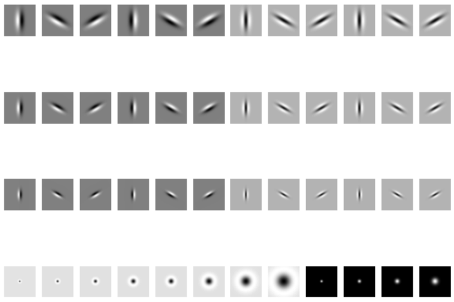

Fig. 3: LML Filters

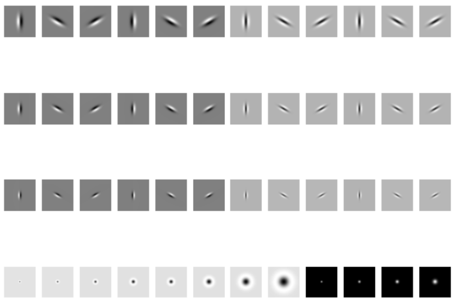

Fig. 4: LMS Filters

3) Gabor Filters: Gabor filters are modeled after the filters found in the human visual system. They are created by combining a Gaussian kernel function with a sinusoidal plane wave.

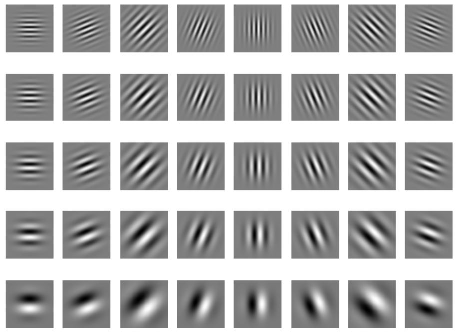

Fig. 5: Gabor Filters

4) Half-Disc Masks: Half-disc masks are binary images that depict half-circles. These masks are crucial as they permit the computation of chi-square distances through filtering, which is much faster than iterating over each pixel's neighborhood and gathering counts for histograms.

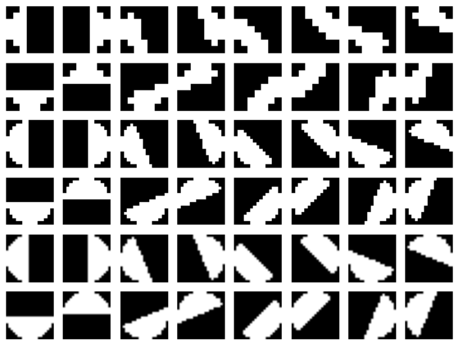

Fig. 6: Half-Disk Filters

B. Mapping

1) Texton Map: Applying each element of the filter bank to an input image produces a vector of filter responses for each pixel. To simplify this representation, each N-dimensional vector is replaced by a discrete texton ID by grouping the filter responses at all pixels in the image into K textons using k-means clustering.

Fig. 7: Texton Map

2) Brightness Map: A brightness map is a representation of the variations in brightness within an image. The process is done by clustering the brightness values of the image using k-means clustering into a specified number of clusters, resulting in a brightness map **B**.

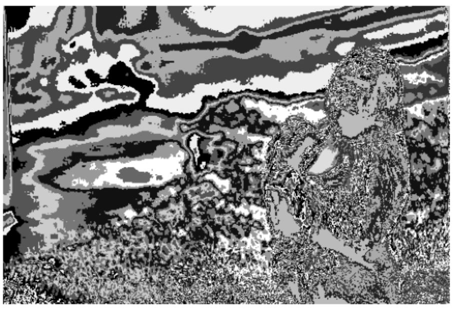

Fig. 8: Brightness Map

3) Color Map: The purpose of a color map is to identify variations in color or chrominance within an image. This is done by using k-means clustering to group the color values of the image into a chosen number of clusters, resulting in a color map **C**. One can also choose to cluster each color channel separately.

Fig. 9: Color Map

C. Gradients

We use the Half-Disc Masks obtained earlier to calculate gradients of the mappings, thus acquiring T_g, B_g, C_g. They encode how much the texture, brightness and color distributions are changing at a pixel. We compare texton, brightness and color distributions with the χ^2 measure. The χ^2 distance is a frequently used metric for comparing two histograms. χ^2 distance between two histograms g and h with the same binning scheme is defined as follows.

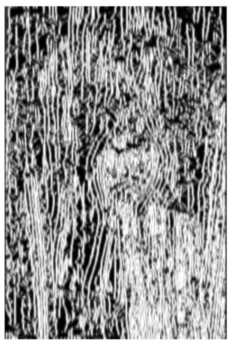

Fig. 10: Texton Gradient

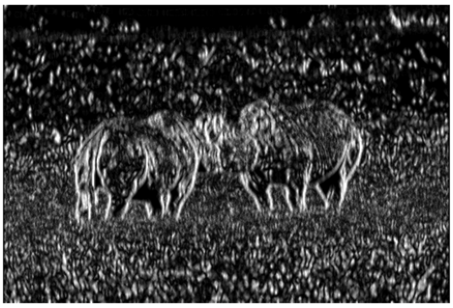

Fig. 11: Brightness Gradient

Fig. 12: Color Gradient

D. Pb-Lite Output

The final step is to combine information from the features with a baseline method (based on Sobel or Canny edge detection or an average of both) using a simple equation

$$PbEdges = \frac{T_g + B_g + C_g}{3} \times (w1 * cannyPb + w2 * sob$$

The magnitude of the features represents the strength of boundaries, thus, the average of the feature vector at location is proportionate to the boundary strength. Though, other methods to combine the features can be investigated for improved performance. A simple approach is to use the element-wise product of the baseline output and the mean feature strength to form the final boundary strength value, which should work well as a starting point.

Fig. 13: Sobel Baseline

Fig. 14: Canny Baseline

Fig. 15: Pb-Lite Output

II. Phase 2

For the purpose of this assignment, three different types of neural networks were trained using the CIFAR10 dataset. The provided starter code was modified to obtain training and testing checkpoints. The input size of the images for these neural networks is 3x32x32.

The SGD optimizer was used as it gave reliably higher accuracy scores than the ADAM optimizer. The cross-entropy loss function was used.

Due to hardware constraints, the models could not be trained for as long as one could've hoped.

A. Simple Network

The simple network consists of 3 successive convolutional layers followed by a linear layer and a softmax layer. It uses the Cross entropy loss function and the SGD optimizer.

```
[652  26  75  24  51  16   9  23 101  23] (0)
[ 36 750  24  20  24   7  22  18  35  64] (1)
[118  21 496  54 104  62  45  67  22  11] (2)
[ 69  27 116 400 103 108  79  43  37  18] (3)
[ 67   9 112  60 522  53  27 131  15   4] (4)
[ 44  13 144 143  72 411  62  82  20   9] (5)
[ 44  34  83  94 110  33 522  34  35  11] (6)
[ 39   3  72  35  77  44  12 700   7  11] (7)
[100  40  24  17  17  12  13  12 743  22] (8)
[ 73 132  31  31  19  19  31  62  36 566] (9)
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
Accuracy: 57.62 %
```

Fig. 16: Simple Model Confusion Matrix

B. Improved Network

The network was improved using the batch normalization method and L2 regukarization to reduce overfitting of the data.

C. ResNet

ResNet-9 is a small convolutional neural network architecture that was introduced in the paper ”Identity Mappings in Deep Residual Networks”. It is a variation of the ResNet architecture that uses fewer layers, making it a smaller and more computationally efficient model. ResNet-9 is commonly used as a building block for larger, more complex models, and is particularly well-suited for image classification tasks. The model's architecture is composed of several residual blocks, which are designed to alleviate the vanishing gradients problem that can occur in deep neural networks. The ResNet-9 model is trained using backpropagation and uses the softmax activation function in the output layer.

The model lacked some training due to hardware issues but gave promising performance.

```
[472  17 111  54 205  24  30  37  21  29] (0)
[ 56 542  31  66  62  19  56  26  36 106] (1)
[104  15 362  96 205  76  29  69  14  30] (2)
[ 57  20  90 392 173  88  79  53  19  29] (3)
[ 74  11  92 102 509  57  26 101  12  16] (4)
[ 32   9 109 212 148 332  60  57  16  25] (5)
[ 75  38  46 183 141  67 342  34  26  48] (6)
[ 74   5  44  68 196  43  36 506   9  19] (7)
[193  50  44 103  86  20  73  19 328  84] (8)
[ 90  87  33  81  33  13  80  30  28 525] (9)
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
Accuracy: 43.1 %
```

Fig. 17: ResNet Confusion Matrix

D. DenseNet

DenseNet is a convolutional neural network architecture that was introduced in the paper ”Densely Connected Convolutional Networks”. It is known for its dense connectivity pattern in which each layer is connected to all other layers in a feed-forward fashion. This dense connectivity pattern allows for efficient information flow and reduces the number of parameters required for the network. DenseNet is particularly well-suited for image classification and segmentation tasks.

Again the model was not able to train enough for it to provide very good results but was able to show promise that it would perform well.

```
[242  41 168  45 165  36  38  81  59 125] (0)
[ 17 604  31  24  13  27  20  45  32 187] (1)
[ 29  34 395  92  92 140  37 106  18  57] (2)
[ 17  24  91 340  72 238  70  86  14  48] (3)
[ 24  25  95 109 293 124  35 250  16  29] (4)
[ 12  13 110 132  39 479  50 129   8  28] (5)
[ 17  45  52 123  34 109 469  84  16  51] (6)
[ 14   6  57  39  51  84  24 666   8  51] (7)
[ 48  67  58  51  33  45  29  33 513 123] (8)
[ 19  87  43  38   7  24  36  57  19 670] (9)
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
Accuracy: 46.71 %
```

Fig. 18: DenseNet Confusion Matrix

III. Conclusion

Although the simple network outperforms the others, it can simply be attributed to a lack of training. The tried and tested models have shown much better performance on datasets with extensive amount of training.