CMSC733
Homework 0

Jo Shoemaker Using (6) Late Days
Computational Linguistics and Information Processing Lab
University of Maryland

***Abstract*—Boundary detection and image recognition are two of the most fundamental tasks in computer vision. In this assignment, we explore a simplification of the Pb-Lite method for boundary detection on ten medium-sized images (around 400 by 300 pixels each), and simple neural architectures for image classification on the CIFAR10 dataset. Unfortunately, time constraints prevented the author from implementing the ResNet, ResNeXt, and DenseNet architectures, and her existing architecture is an utter failure for reasons that remain as future work.**

I. Phase 1: Pb-Lite

Pb-Lite is a boundary detection method that determines the probability of a boundary at each pixel based on three sources of information: color change, texture change, and brightness change. Where all three are changing dramatically, there is a high probability of a boundary. Detecting these changes is nontrivial: one first needs to have a representation of the image composed of discrete values for textures, colors, and brightnesses. Color and brightness for a pixel are already well-defined by its RGB and greyscale values, and so each pixel can be assigned a discrete color and brightness bin based on k-means clustering of these values. Determining texture is a little more difficult. The Pb-Lite solution is to characterize texture at a pixel as a vector of results of convolving the greyscale image with various filters. These vectors are then binned and mapped to an index by k-means clustering. Finally, the by-pixel "gradients" in the color, brightness, and texture domains are determined by comparisons of opposite pairs of half-disc masks convolved with the image at different scales and orientations.

In this assignment, we tested out the Pb-Lite algorithm on ten images that were around 400×300 pixels each. The sections below discuss each step of the algorithm in more detail.

A. Color and Brightness Classification

For color binning, raw RGB pixel values across all ten images were used to train k-means clustering for k=25. Figure 1 illustrates the resulting gradient-in-color maps for each of the ten images. For brightness binning, the greyscale pixel values across all ten images were used to train k-means clustering for k=16. Figure 2 illustrates the resulting gradient-in-brightness maps for each of the ten images.

B. Texture Classification

Unlike color and brightness, texture is not defined on a by-pixel basis. However, by convolving our greyscale images with filters that highlight value changes across subregions of the image, the resulting convolutions assign each pixel a value that characterizes its surroundings. The 147 filters generated for this purpose are at varying scales and orientations. 24 of these are square derivative of gaussian kernels (Figure 3), 75 come from the Leung-Malik filter set (Figure 4), and 48 are Gabor Filters (Figure 5).

The resulting 147-value vectors for each pixel were assigned to indices determined by k-means clustering with k=64. The discrete-texture representations of each image are shown in Figure 6, and the gradients in these textures in Figure 7.

C. Determining Gradients

To determine the by-pixel gradients in our color, brightness, and texture spaces, we performed chi-squared comparisons of convolutions with opposite half-disc masks for each discrete color, brightness, and texture value. The chi-squared results for each half-disc mask pair (shown in Figure 8) were averaged together to obtain the final gradient result.

D. Results and Commentary

The final output of Pb-Lite (shown for our ten images in Figure 9) is determined by averaging the three gradient values at each pixel and multiplying by the average Canny and Sobel baseline boundary likelihood for that position. The results clearly catch more boundaries than the Sobel baseline, which is unsurprising given that a Sobel filter at a single orientation can't pick up on boundaries that may lie at arbitrary orientations. The Canny baseline is very comparable to my Pb-Lite output, despite considering only grayscale derivative of gaussian filter results. Instead of leveraging color and texture information to eliminate spurious boundary candidates caused by noisy brightness gradients, the Canny method uses a multi-stage weed-out process. It's not surprising that the strongest boundaries that survive this weed-out process are very similar to those found by Pb-Lite, since strong brightness changes are often correlated with color and (larger-scale) texture changes.

There are a few things I could think of that make the Pb-Lite algorithm less effective than it could be. For one, assigning arbitrary index values to ranges of color/texture/brightness is rather silly, since some bins are more similar than others and may consistently appear next to each other in images. For

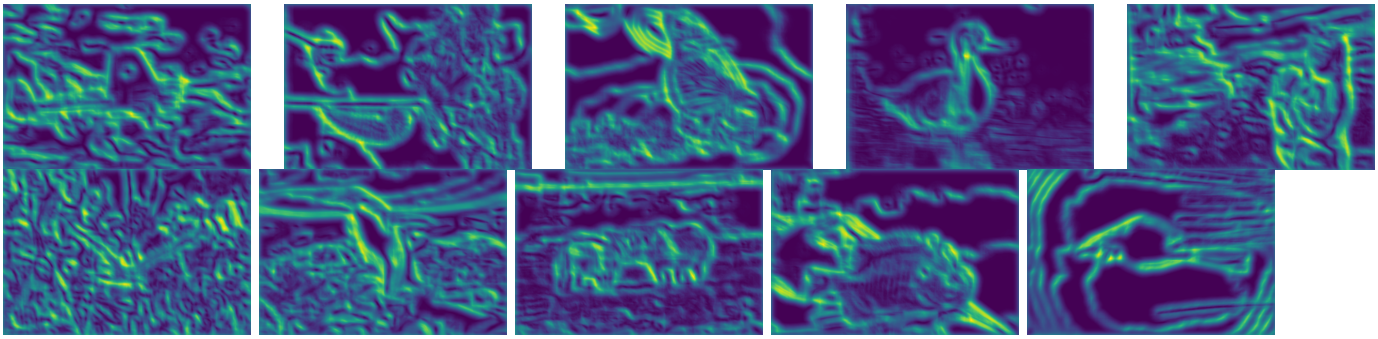

Fig. 1. Figure 1: Color gradients for each image.

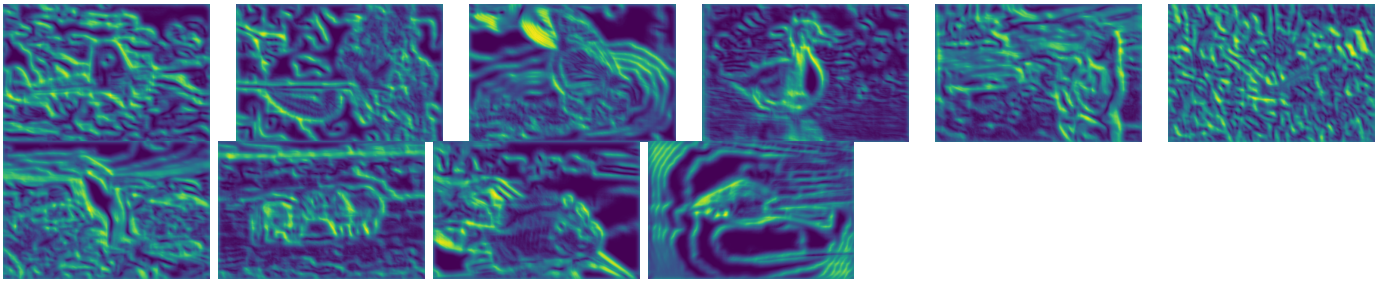

Fig. 2. Figure 2: Brightness gradients for each image.

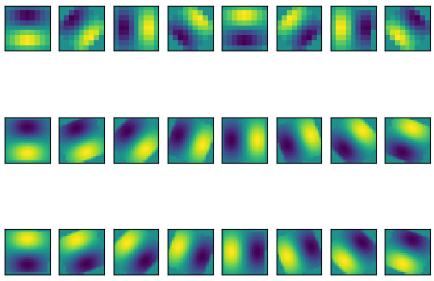

Fig. 3. Figure 3: Derivative of Gaussian filters at nine- and 21-pixel scales and eight and 16 rotations respectively.

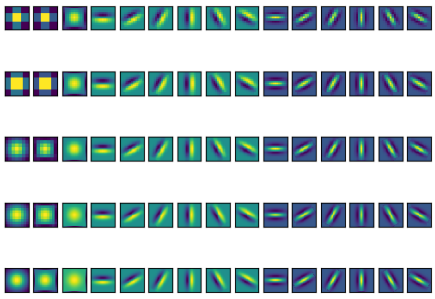

Fig. 4. Figure 4: The Leung-Malik filter set for scales 1, $\sqrt{2}$, 2, $2\sqrt{2}$, and 4.

example, if a region that is fairly consistent in texture steadily changes from light blue to dark blue, that should be a cue that both the darker blue shade and the lighter one belong in the same bin or at least in 'close' bins. There is also no reason one couldn't apply the Canny weed-out procedure to Pb-Lite boundaries as well to eliminate noise.

II. Phase 2: Neural Architectures

I am not a vision researcher, but it is my understanding that deep convolutional networks are now the default for all image processing tasks. Convolutional neural networks (CNNs) are able to learn filters which behave similarly to the hand-spun filters in Phase 1 by characterizing meaningful sub-patterns within images on a per-pixel basis. By stacking these layers—and usually downsampling to lessen the noise with each stack—the network can learn more and more complex patterns which begin to resemble abstract characteristics of recognizable entities (e.g., the eye and nose pattern of a dog's face).

In this assignment, we were tasked with classifying the CIFAR10 dataset of 50k 32×32 pixel images into ten classes. Preliminary results of an attempted model are discussed below.

A. An Initial Model

My initial model (Figure 10) is based very heavily off of the one from the official TensorFlow CIFAR10 tutorial. The network consists of three convolutional layers of 64 filters each followed by three fully-connected layers with 200, 90, and 10

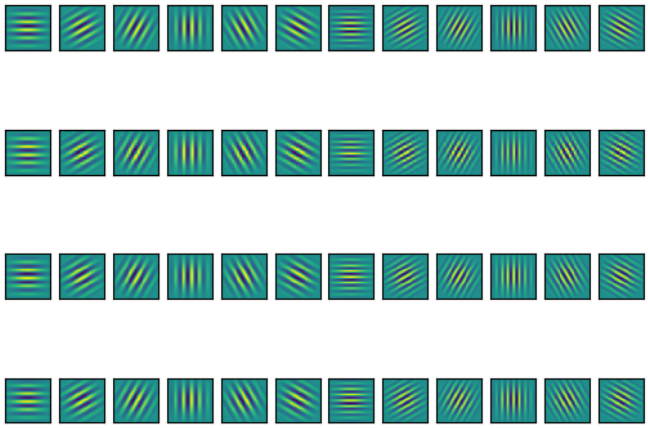

Fig. 5. Figure 5: Gabor filters at 35, 51, 73, and 99-pixel scales and two different wave frequencies with six orientations each.

nodes each. In hopes of simplifying the task of edge detection early on, I separated the first convolutional layer into two such that 36 filters were applied to just the luma channel and 36 were applied to the two chrominance channels. The last two convolutional layers are immediately followed by max pooling and downsampling. Each layer except the first is immediately preceded by a RELU activation.

B. Training

The model was trained to minimize cross-entropy loss according to the Adam Optimizer at a learning rate of 0.1, $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-8$. The model trained for 20 epochs with a batch size of 125 images. For preprocessing, images were converted to the luma-chrominance color space and standardized to have pixel values in the range $[-1, 1]$. The model never achieved performance significantly above chance during any epoch (see Figure 11), so I didn't bother with running it on the test set. Other learning rates (0.05, 0.01, 0.001) and batch sizes (10,50) were attempted to no avail.

ACKNOWLEDGMENT

The authors would like to thank the heroic but anonymous author(s) of the TensorFlow and NumPy documentation pages, which were integral to producing this project. Pointedly unacknowledged is the author of the OpenCV documentation, who was incorrect on multiple counts and should be fired.

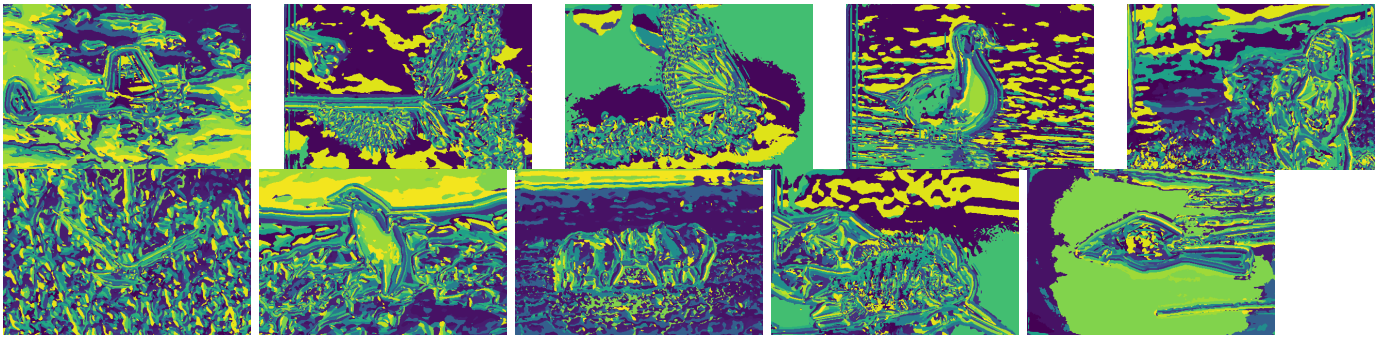

Fig. 6. Figure 6: Discrete-texture representations of each image.

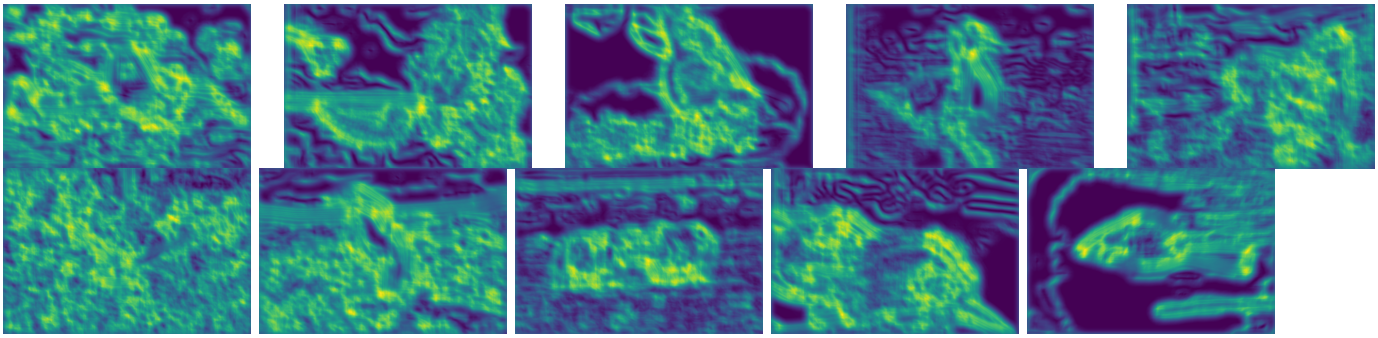

Fig. 7. Figure 7: Texture gradients for each image.

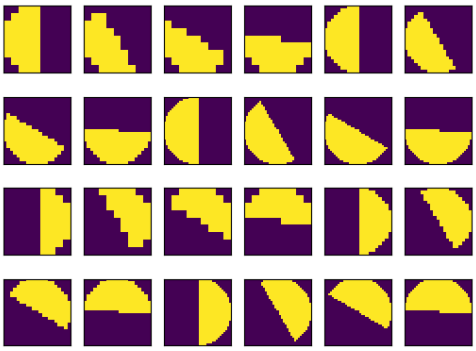

Fig. 8. Figure 8: Half-Disc masks at 9, 21, and 35-pixel scales and four orientations. The left-side masks for all three scales are shown before and above their right side counterparts.

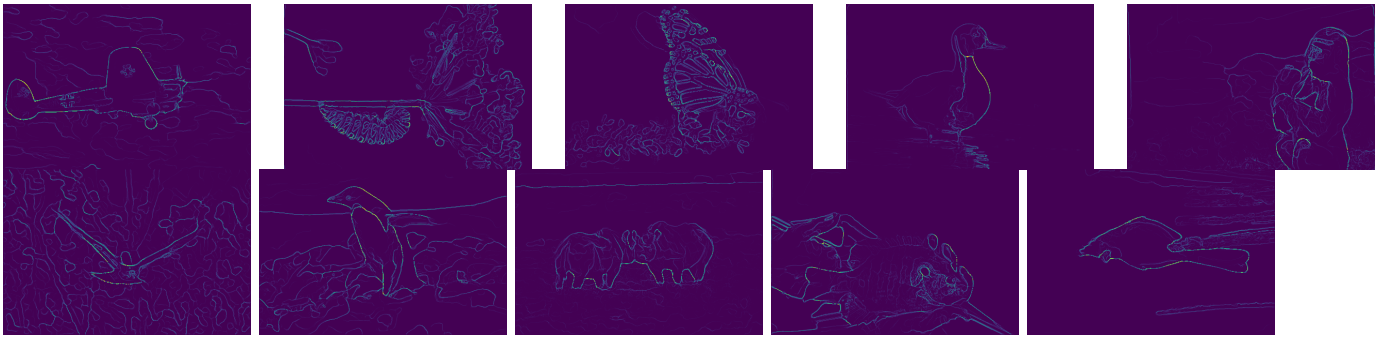

Fig. 9. Figure 9: Pb-Lite output for each image.

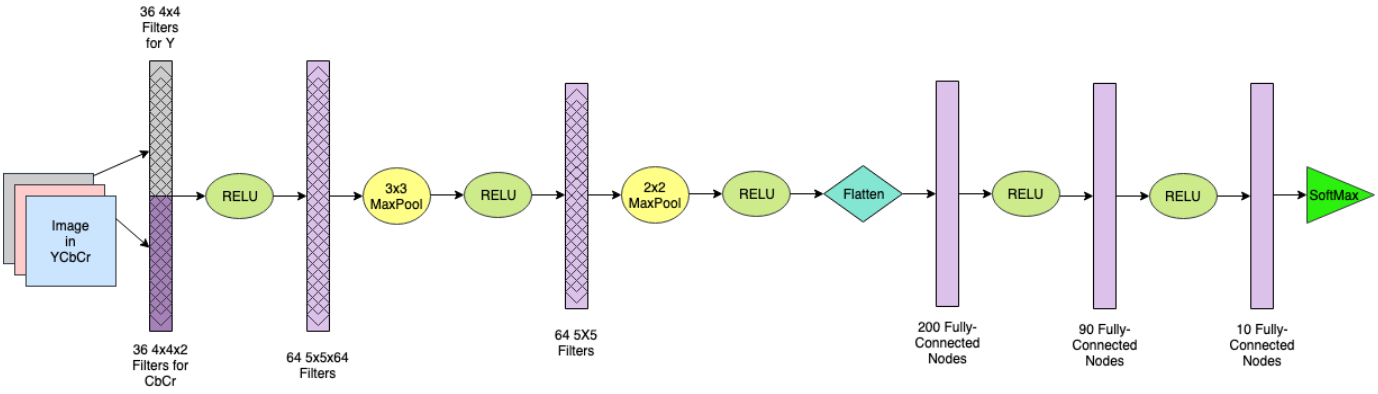

Fig. 10. Figure 10: Initial Architecture

Fig. 11. Figure 11: Accuracy on the test set by epoch (left), cross-entropy loss by epoch (right).