Homework 0 - Alohomora! Submission

Abhishek Kathpal
CMSC733
UID:114852373
Email: akathpal@umd.edu
USING 3 LATE DAYS

***Abstract*—This homework has two phases. Phase 1 is focussed on implementation of pb (probability of boundary) algorithm using the texture, brightness and color information combined with the canny and sobel baselines. Phase 2 is implementation of multiple deep learning architectures: ResNet, ResNeXt and DenseNet on CIFAR10 Dataset. Few approaches for improving train and test dataset accuracies are discussed and implemented.**

I. Phase1

A. Overview

The goal of this part of project is to compute the edges in a given image. The traditional approaches for solving this problem are Canny Edge and Sobel Edge Detector. Sobel computes the differences in intensity accross neighboring pixels after converting into grayscale image. A predined value is used to find the edges sharper than that value in the image. Canny determines whether pixel belongs to edge or not by checking the similarity between the gradients at that pixel and the neighboring pixels. Both of these algorithms only takes change in intensity into account.

Pb Algorithm considers texture, brightness and color changes to detect the per pixel probability of being a point on edge in the image.

The pipeline is given by following figure:

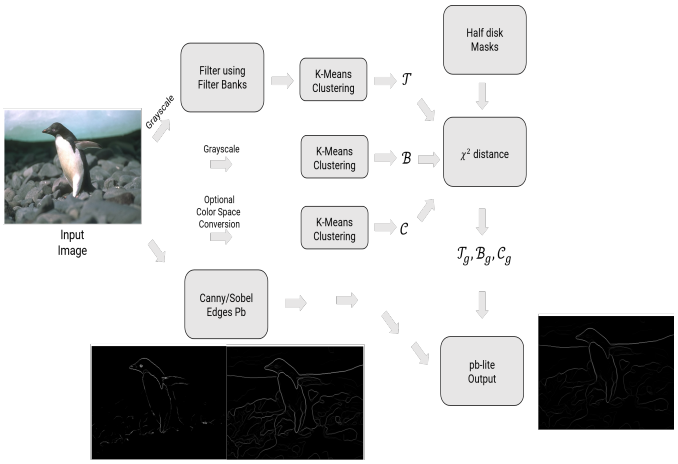

Fig. 1. Pb Algorithm pipeline

The pb algorithm can be implemented using the following steps: 1. Generating filter banks using Oriented DoG Filters, Gabor Filters and Leung-Malik Filters. 2. Computing the Texton Map using the filter bank and KMeans CLustering 3. Using Half-disc masks to compute the gradient maps of texture, brightness and color. 4. Using the given Sobel and Canny baselines to have a rough idea of edge. 5. Computing the per pixel probability of pixel belonging to an edge in image.

B. Filter Banks

Three filter banks-Gabor, DoG and Leung-Malik have been used in this approach to detect textures in the given input image. These filter banks have filters at different scales to detect scale change and different orientations to compute textures in different orientations. The output from these filter banks are shown below:

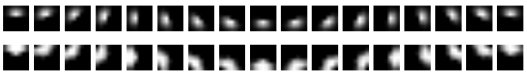

Fig. 2. DoGFilterBank

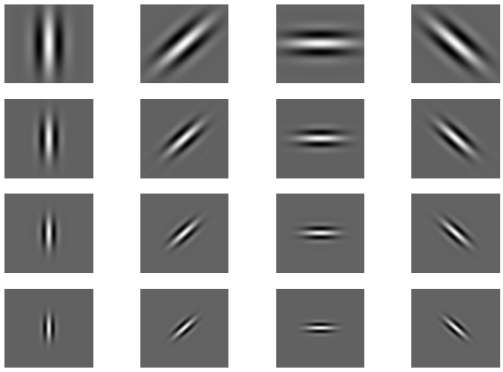

Fig. 3. GaborFilterBank

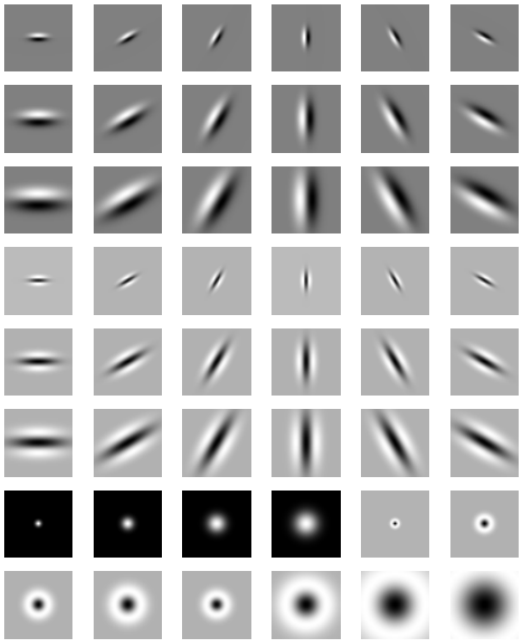

Fig. 4. LMFilterBank

These filter banks are used to found the texton , brightness and color maps. These maps for penguin image is shown in figures below.

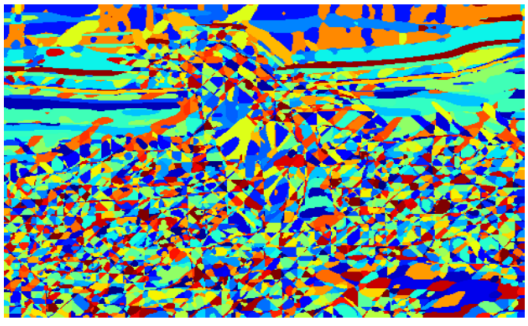

Fig. 5. TextonMap

Fig. 6. BrightnessMap

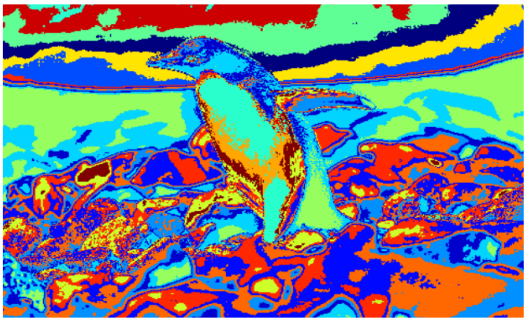

Fig. 7. ColorMap

C. Half Disc Masks

Half-Disc Masks are used to compute the gradients of the computed texture, brightness and color maps. These will help us compute the chi-square distances using less number of computations. The half discs are shown below:

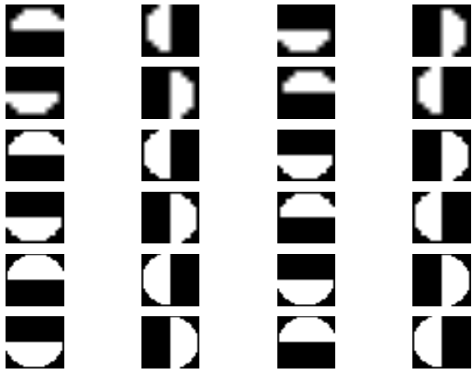

Fig. 8. HalfDiscMasks

These gradient maps will tell the change in distributions of texture, brightness and color at a pixel. These are shown in the figures below.

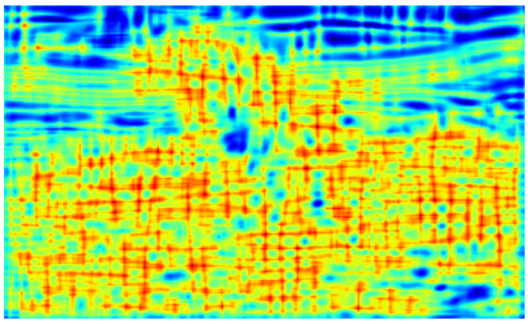

Fig. 9. TextonGradient

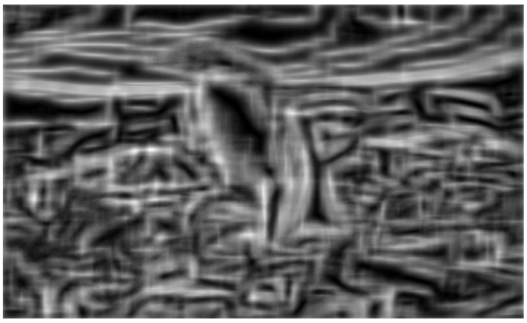

Fig. 10. BrightnessGradient

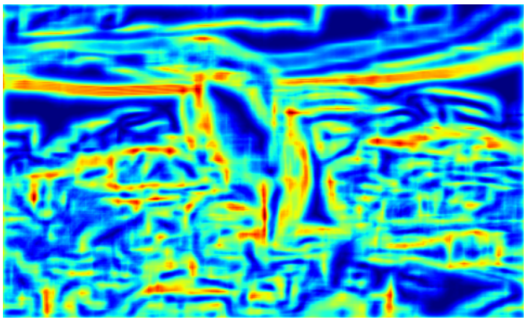

Fig. 11. ColorGradient

D. Pb-lite Output

To finally compute the output from the algorithm, the following equations is used:

$$PbEdges = \frac{T_g + B_g + C_g}{3}.*(w_1 * Canny + w_2 * Sobel) \quad (1)$$

Sobel and Canny baselines shown are being used for computing the edges in the original image.

Fig. 12. CannyBaseline

Fig. 13. SobelBaseline

The final output is shown in the figure below:

Fig. 14. PbLiteOutput

E. Advantages

Pb output is much better in finding boundaries of objects qualitatively. From the algorithmic point of view, this takes into account the texture and color details into account as well in comparison to just intensity changes in case of Sobel and Canny. One downside of Pb lite is that if fails to detect the edges if the edges are not in both canny and sobel baselines, as this algorithm filter the poor pixel candidates for the edges.

II. PHASE2

1) Overview: In this part of project, the goal is to implement different deep learning architectures to classify different objects based on CIFAR10 dataset. THe first task is to choose loss function and optimizer function. Cross Entropy with softmax is used a loss function, this is generally the choice for classification tasks. I have used Adaptive Moment Estimation(Adam). It is better than other optimizers like RMS prop , Adagrad as Adam decaying average of past squared and past gradients, it generally works better than other adaptive learning- method based algorithms. In this Phase, I have first implemented the basic CNN architectures using just convolutional and fully connected layers. Then I studied the architectures of ResNet, ResNext and DenseNet and tried to implement these architectures to improve the accuracy. I was only able to successfully execute ResNet architecture. The architectures are explained in the section below.

2) Different Architectures: The first architecture is simple CNN. I have used for this architecture 3 convolutional layers with max pooling, batch normalization and 4 fully connected layers. I was getting an accuracy of around 65% on Test Dataset. I ran the code for 20 epochs with 64 batch size. In the fig. below, the loss and accuracy per epoch are shown.

I have also plotted the confusion matrix heat map for the Test Dataset.

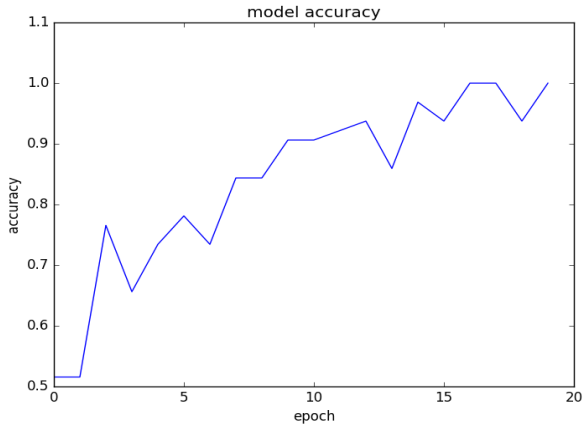

Fig. 15. Accuracy plot vs Epochs

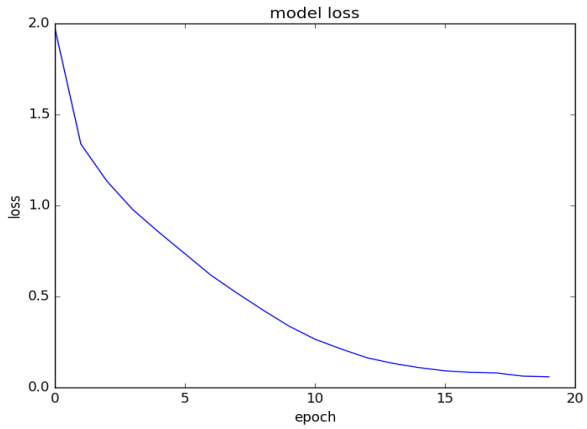

Fig. 16. Loss plot vs epochs

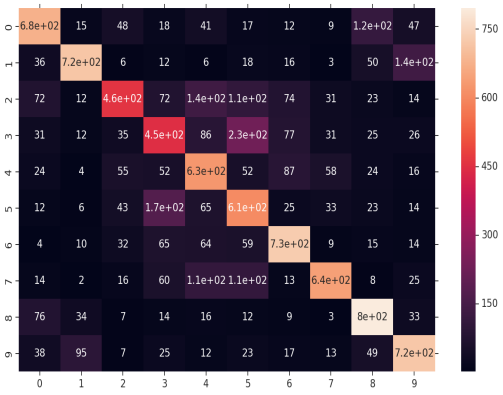

Fig. 17. Confusion Matrix

To improve the above results, another architectures have been used. All of these architectures uses skip connections. ResNet uses residual blcoks which helps in solving the problem of vanishing gradient. The architecture of ResNet is shown in the figure.

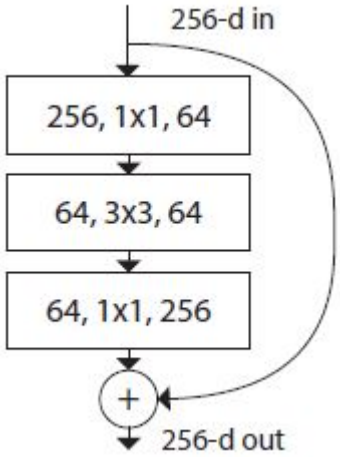

Fig. 18. ResNet Architecture

I have implemented ResNet architecture with only 3 residual blocks and few fully connected layers in the end with softmax function. The accuracy and loss plot per epoch with this small ResNet architecture is shown below:

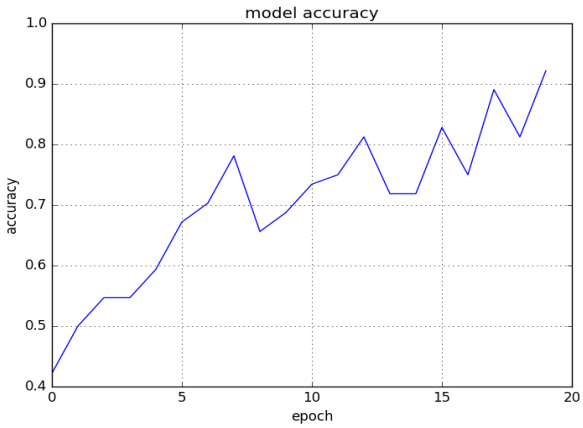

Fig. 19. Accuracy plot vs Epochs

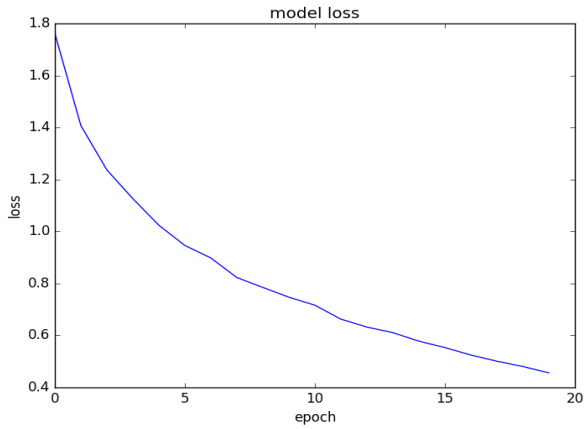

Fig. 20. Loss plot vs epochs

Another modified version of ResNet is ResNeXT architecture which introduced the concept of cardinality which refers to how many times the input is split. I was not able to implement this architecture. The architecture is shown below:

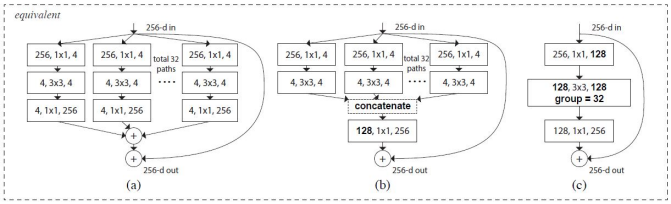

Fig. 21. ResNext Architecture

Dense Net architecture is shown below. In this architecture, skip connections are used in different way, the output from one layer is added to all the next layers output and feeded as input.

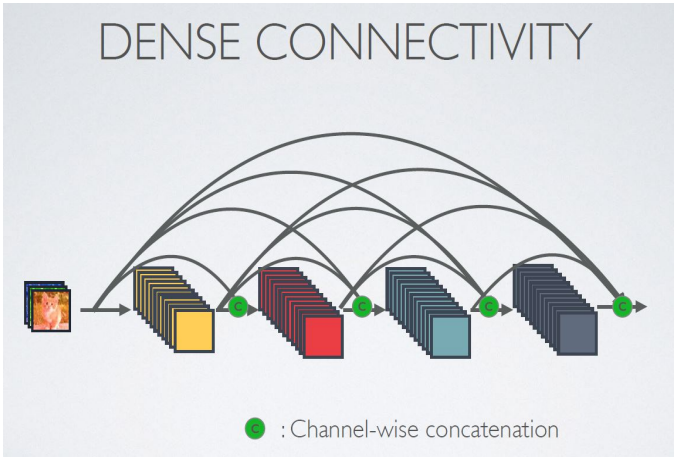

Fig. 22. Dense Net Architecture

The comparison between the architecture of CNN, ResNet and DenseNet is shown in fig. below

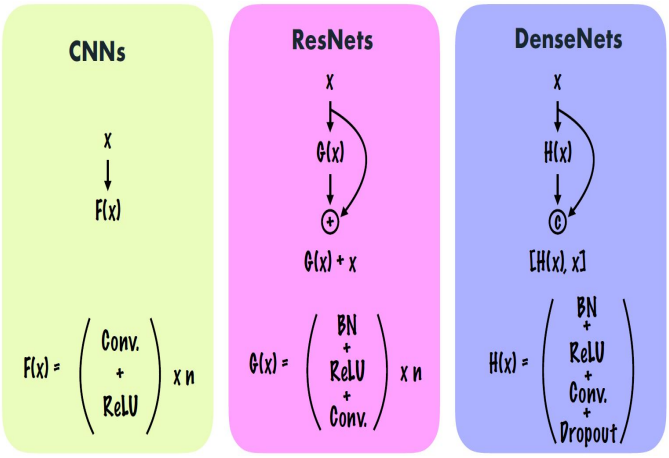

Fig. 23. Comparison

REFERENCES

[1] Arbelaez, Pablo, et al. "Contour detection and hierarchical image segmentation." IEEE transactions on pattern analysis and machine intelligence 33.5 (2011): 898-916.

[2] http://cs.brown.edu/courses/cs143/2011/proj2/

[3] http://www.robots.ox.ac.uk/ vgg/research/texclass/filters.html

[4] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[5] https://github.com/taki0112/Densenet-Tensorflow

[6] Xie, Saining, et al. "Aggregated residual transformations for deep neural networks." Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on. IEEE, 2017.