Homework 0 - Alohomora!

Aruna Baijal
USING 1 LATE DAY
UID:116130110
Email: abaijal@umd.edu

***Abstract*—We begin this course by working our way through the core concepts of computer vision and deep learning. We will generate an edge detection algorithm that will significantly outperform well regarded Canny and Sobel edge detectors. In the second phase, we develop our own Neural Networks and implement a few well-known architectures. The aim is to understand how each network is created and can be manipulated.**

I. Phase 1 - Shake My Boundary

In this section we developed an edge detection algorithm that uses brightness, texture and color across multiple scales. Hence, our algorithm outperforms classical edge detection techniques like Canny and Sobel. Our algorithm is based on pb (probability of boundary) boundary detection algorithm. The algorithm has 4 main steps - Filter bank generation, Texton, Brightness and Color map generation, Gradient generation for each map, and finally Hadamard product with Canny and Sobel baseline.

A. Filter Banks

First we generate a filter bank of Oriented Derivative of Gaussian filters, Leung-Malik Filters and Gabor Filters. Each type of filter has multiple scales and orientation. Using multiple filters on our images help us measure texture properties better and to aggregate the brightness and texture distributions locally.
The three filter banks are as shown below. The DoG filter has 2 scales(5, 11) and 16 orientations. The LML filter has 4 scales($\sqrt{2}, 2, 2\sqrt{2}, 4$) and 16 orientations. The Gabor filter has 2 scales(5,11) and 16 orientations.

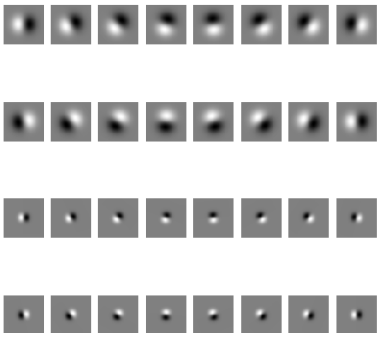

Fig. 1: DoG Filter

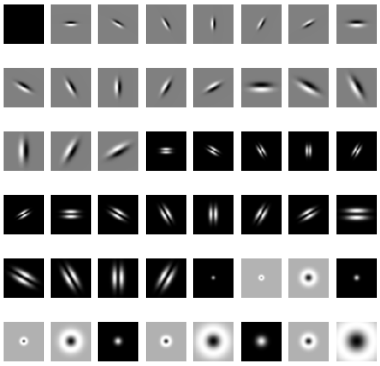

Fig. 2: LML Filter

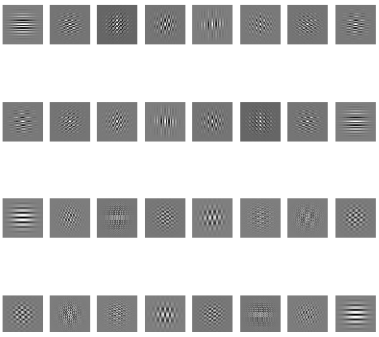

Fig. 3: Gabor Filter

B. Texton Map, Brightness Map, Color Map

The Texton map is generated by applying all filters in the filter bank to the image. The filter responses at each pixel is then represented by a single cluster value (out of 64 clusters) found out using K-means.
Brightness map clusters pixels to 16 clusters using K-means. according to the brightness value of each pixel.
Color map clusters the pixels into one of 16 bins based on its RGB value using K-means.

C. Texture, Brightness and Color Gradients

We find the gradients by calculating the chi-square distance between two histograms. These histograms are generated by filtering the maps with two opposite half disc filters. The half disc filter bank is made up of 3 radius discs (5, 10 and 15) and 8 orientations. Each half disc has an opposite disc as well. The half disc filter bank is shown below.

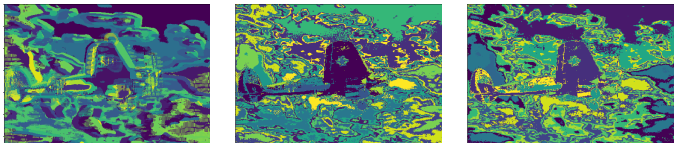

Fig. 4: $\mathcal{T}, \mathcal{B}, \mathcal{C}$ of image 1

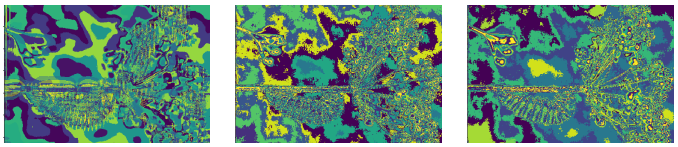

Fig. 5: $\mathcal{T}, \mathcal{B}, \mathcal{C}$ of image 2

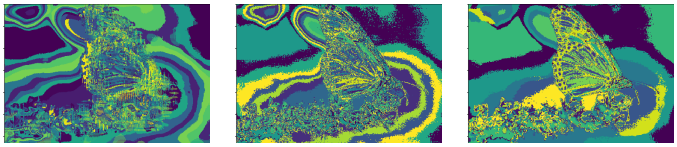

Fig. 6: $\mathcal{T}, \mathcal{B}, \mathcal{C}$ of image 3

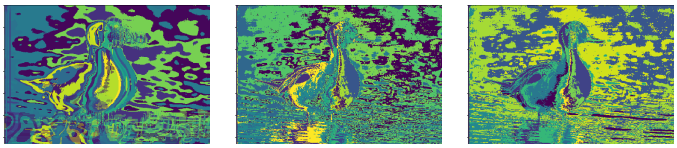

Fig. 7: $\mathcal{T}, \mathcal{B}, \mathcal{C}$ of image 4

The gradient maps generated using these half discs are as shown.

D. Edge detection

To detect edges we take the Hamdamard product of average of the threee gradients with the weighted sum of Canny and Sobel baselines. As we can see in the final images, our pb-lite edge detection algorithm has worked better than both Canny and Sobel algorithms. Pb-lite has edges that weren't detected by Sobel and sharper edges than Canny. But increasing the weight of Canny sum we can increase the brightness of the background edges.

II. Phase 2 - Deep Dive on Deep Learning

A. My First Neural Network

I used a simple CNN architecture with two convolution layers. A ReLU layer after each convolution layer. A maxPoolingLayer after each ReLU layer. Training runs for 20 Epochs and a batch size of 256. Total number of images used for training are 50000. This model has 273706 parameters and a test accuracy rate of 10.25%. In addition, I used an Adam optimizer with a learning rate of $1e-3$ The model architecture, test and train accuracy, loss over epochs and confusion matrix are given below. We can deduce that the model has overfit to the training data.

B. Modified Network

To the simple CNN network, I added batch normalization and drop out layers. The network was trained over 20 epochs and batch size of 256. The large batch size resulted in the reduction of accuracy. Due to the lack of time and resources I could not retrain the model after tweaking parameters. The number of parameters for the model are 99530 and a test accuracy of merely 10.04%. Another useful addition to the algorithm would be data manipulation for instance flipping pictures along the vertical axis at random. The architecture and results are shown below.

C. ResNet

The architecture I used for ResNet consisted of 2 residual blocks and a kernel size of 3. Training ran for 20 epochs and a batch size of 256. The number of parameters in the model are 4433418. The test accuracy was 35.06%. I did not get a chance to play with the parameters to get better results due to lack of time and resources. One change that I can make is to use Global Average Pooling instead of Max Pooling to get better results. Other results for this model are shown below.

D. DenseNet

Implemented DenseNet with the architecture shown below. Model was trained over 20 epochs with a batch size of 256. The results were not at all satisfactory. As future work, need to fine tune the network with appropriate parameters and understand impact of each parameter.

E. Analysis

Apart from changing parameters, we should implement data augmentation and manipulation as well to improve network training. Choosing an optimum batch size is critical, too small results in the network taking too long to train, too large reduces accuracy of the network. Further understanding of dropout layer and parameters like drop rate required.

III. Conclusion

The effects of different filters on image manipulation is quite evident with this project. Leveraging established edge detection techniques Canny and Sobel to further improve the edges detected by Pb-Lite ensures a better result. The project utilizes all aspects of the image - color, texture and brightness. I learnt a few key points with regard to deep learning. There are numerous deep learning techniques available. Each architecture works best for a unique combination of parameters dependent on the data and environment. A larger/deeper network doesn't mean better accuracy.
This has been a good start into the field of Computer Vision and Deep Learning, that highlighted the areas I need to improve my knowledge. Through the duration of this course,

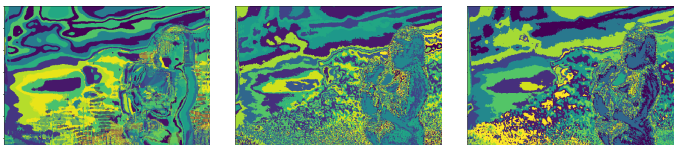

Fig. 8: $\mathcal{T}, \mathcal{B}, \mathcal{C}$ of image 5

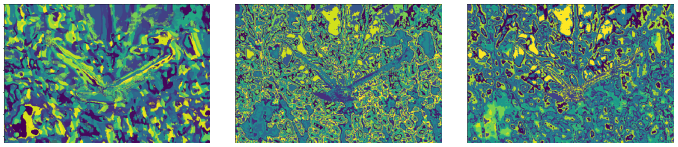

Fig. 9: $\mathcal{T}, \mathcal{B}, \mathcal{C}$ of image 6

I look to gain what was lacking in this project and work successfully through other projects as well.

IV. References

1) Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, "Densely Connected Convolutional Networks":CVPR 2017
2) Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition": arXiv:1512.03385 [cs.CV]

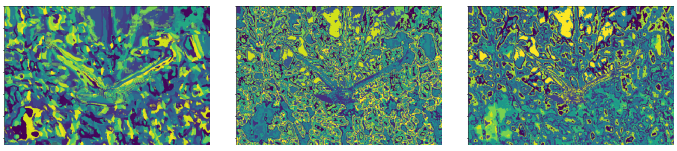

Fig. 10: $\mathcal{T}, \mathcal{B}, \mathcal{C}$ of image 6

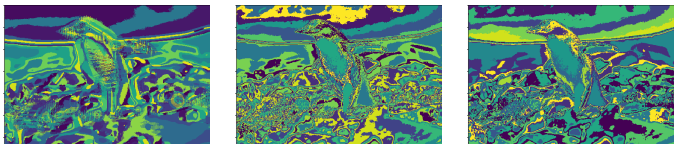

Fig. 11: $\mathcal{T}, \mathcal{B}, \mathcal{C}$ of image 7

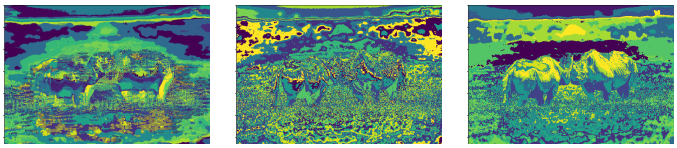

Fig. 12: $\mathcal{T}, \mathcal{B}, \mathcal{C}$ of image 8

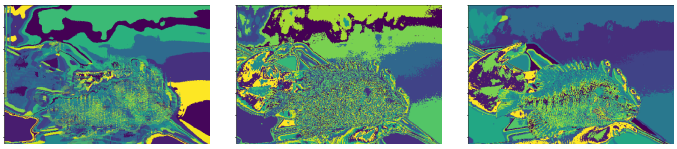

Fig. 13: $\mathcal{T}, \mathcal{B}, \mathcal{C}$ of image 9

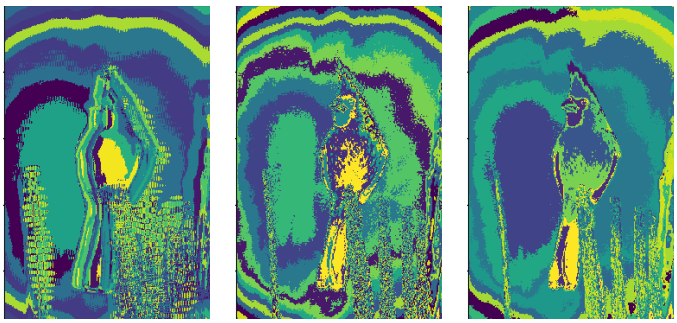

Fig. 14: $\mathcal{T}, \mathcal{B}, \mathcal{C}$ of image 10

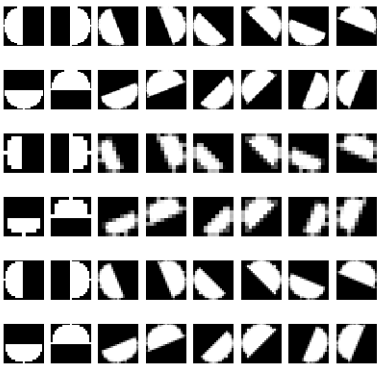

Fig. 15: Half disc Filter

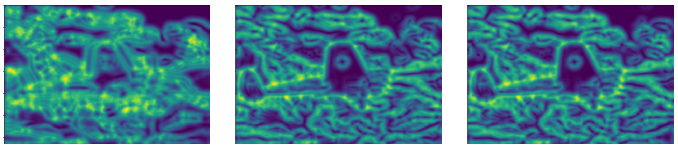

Fig. 16: $\mathcal{T}_g, \mathcal{B}_g, \mathcal{C}_g$ of image 1

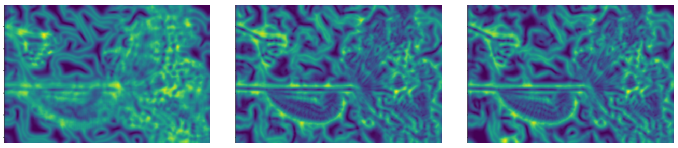

Fig. 17: $\mathcal{T}_g, \mathcal{B}_g, \mathcal{C}_g$ of image 2

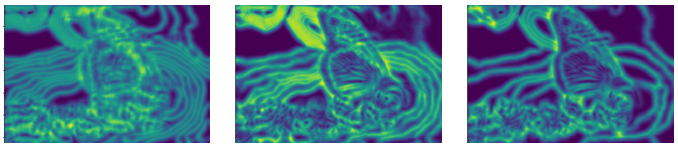

Fig. 18: $\mathcal{T}_g, \mathcal{B}_g, \mathcal{C}_g$ of image 3

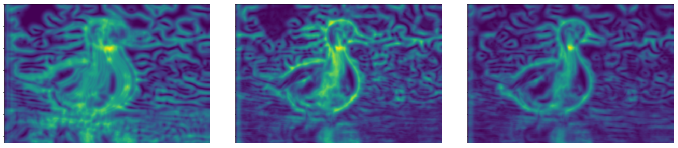

Fig. 19: $\mathcal{T}_g, \mathcal{B}_g, \mathcal{C}_g$ of image 4

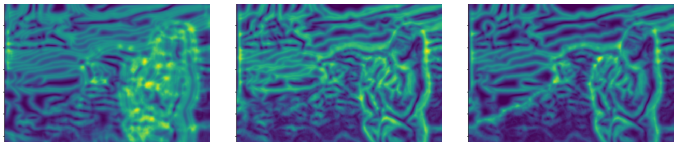

Fig. 20: $\mathcal{T}_g, \mathcal{B}_g, \mathcal{C}_g$ of image 5

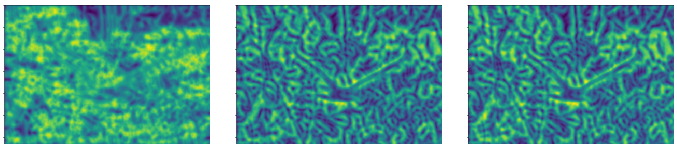

Fig. 21: $\mathcal{T}_g, \mathcal{B}_g, \mathcal{C}_g$ of image 6

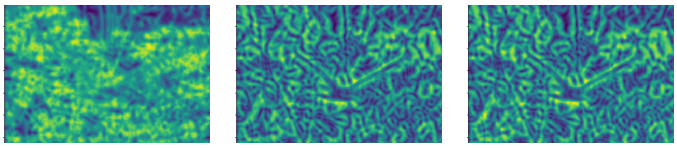

Fig. 22: $\mathcal{T}_g, \mathcal{B}_g, \mathcal{C}_g$ of image 6

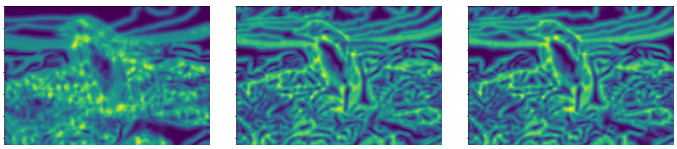

Fig. 23: $\mathcal{T}_g, \mathcal{B}_g, \mathcal{C}_g$ of image 7

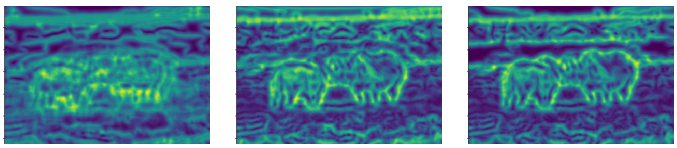

Fig. 24: $\mathcal{T}_g, \mathcal{B}_g, \mathcal{C}_g$ of image 8

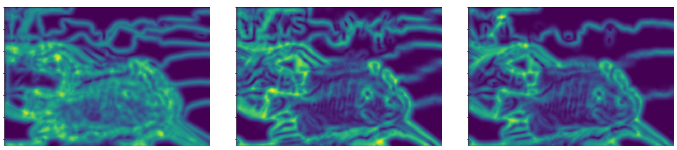

Fig. 25: $\mathcal{T}_g, \mathcal{B}_g, \mathcal{C}_g$ of image 9

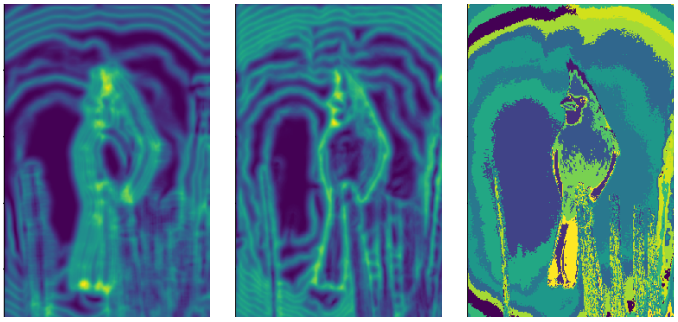

Fig. 26: $\mathcal{T}_g, \mathcal{B}_g, \mathcal{C}_g$ of image 10

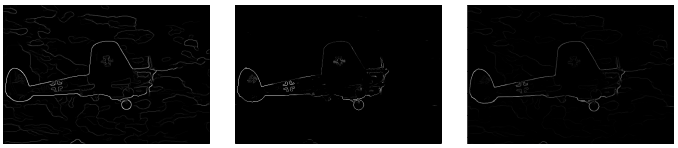

Fig. 27: Canny, Sobel, Pb-Lite output of image 1

Fig. 28: Canny, Sobel, Pb-Lite output of image 2

Fig. 29: Canny, Sobel, Pb-Lite output of image 3

Fig. 30: Canny, Sobel, Pb-Lite output of image 4

Fig. 31: Canny, Sobel, Pb-Lite output of image 5

Fig. 32: Canny, Sobel, Pb-Lite output of image 6

Fig. 33: Canny, Sobel, Pb-Lite output of image 6

Fig. 34: Canny, Sobel, Pb-Lite output of image 7

Fig. 35: Canny, Sobel, Pb-Lite output of image 8

Fig. 36: Canny, Sobel, Pb-Lite output of image 9

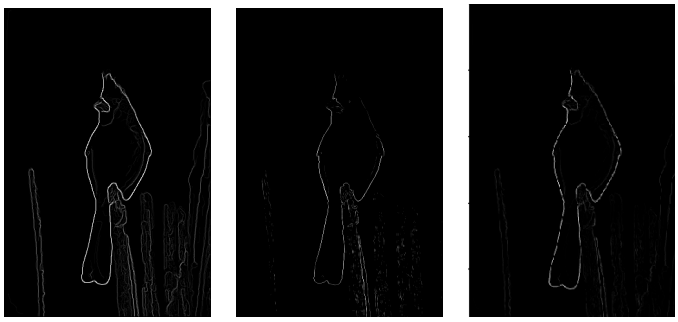

Fig. 37: Canny, Sobel, Pb-Lite output of image 10

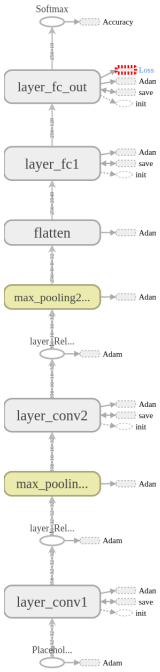

Fig. 38: Architecture of simple CNN

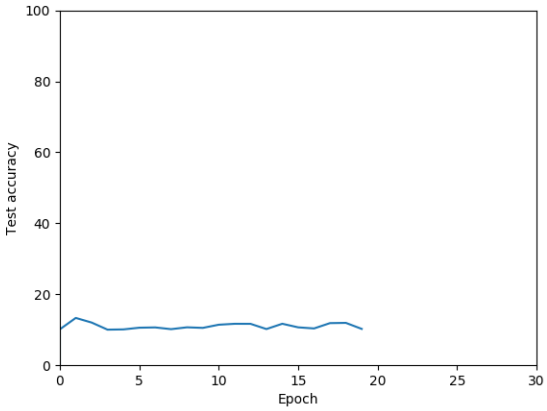

Fig. 40: Accuracy during testing of simple CNN

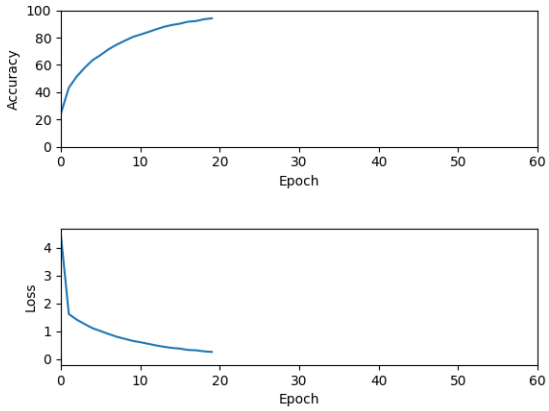

Fig. 39: Accuracy and Loss during training of simple CNN

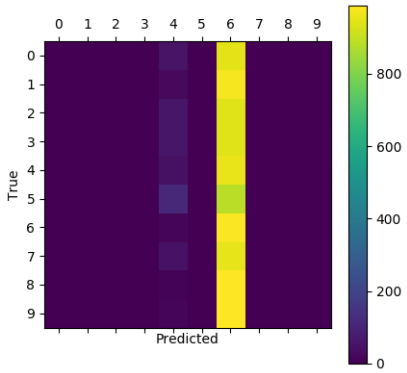

Fig. 41: Confusion matrix during testing of simple CNN

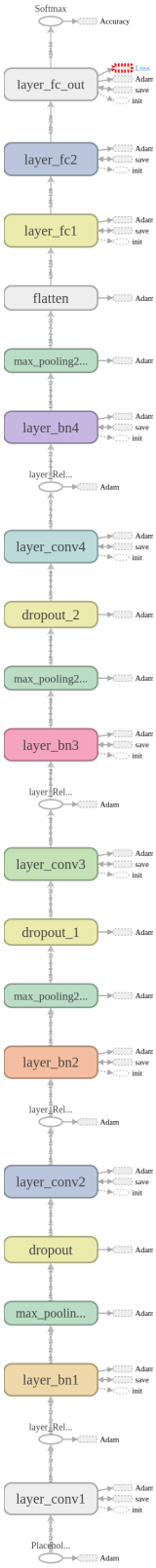

Fig. 42: Architecture of Modified network

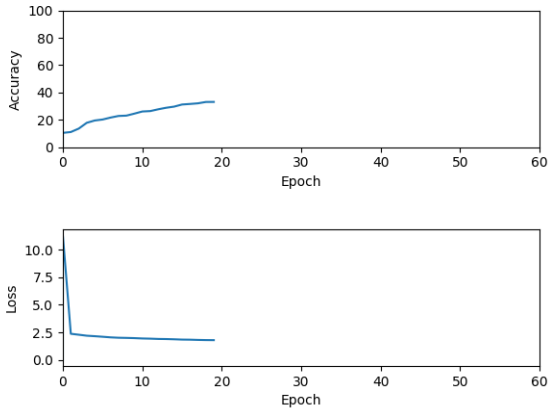

Fig. 43: Accuracy and Loss during training of Modified network

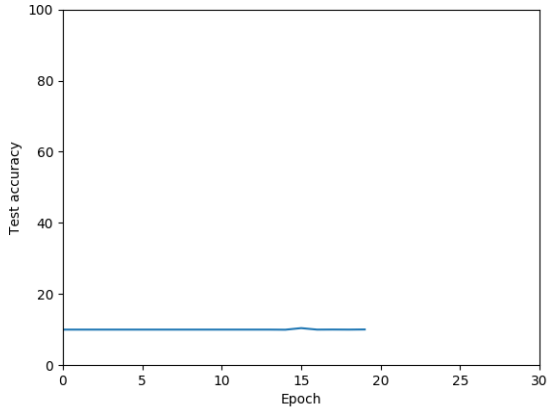

Fig. 44: Accuracy during testing of Modified network

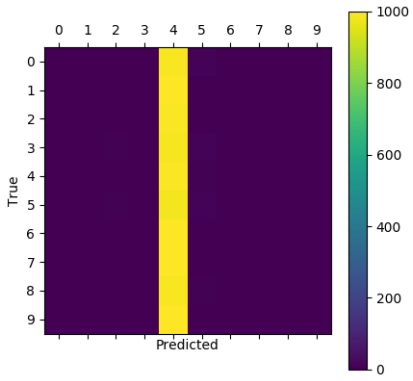

Fig. 45: Confusion matrix during testing of Modified network

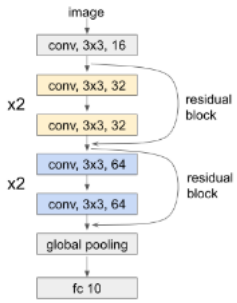

Fig. 46: Architecture of ResNet

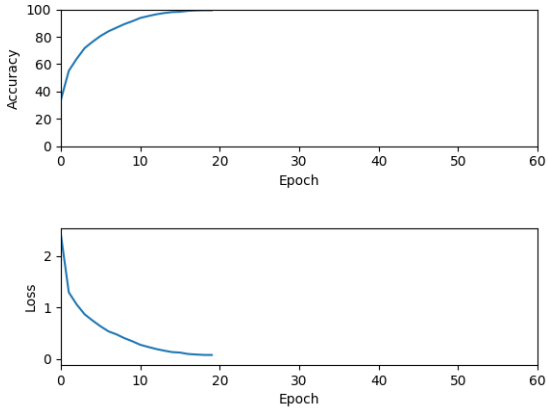

Fig. 47: Accuracy and Loss during training of ResNet

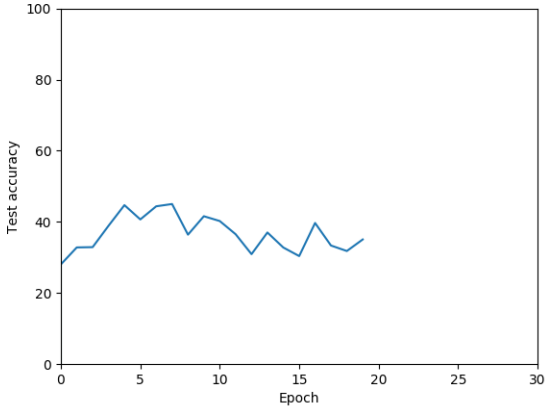

Fig. 48: Accuracy during testing of ResNet

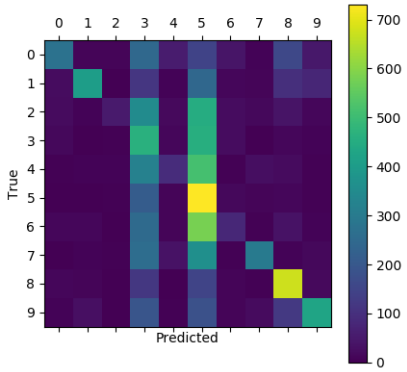

Fig. 49: Confusion matrix during testing of ResNet

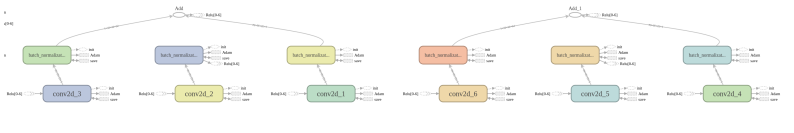

Fig. 50: Architecture of DenseNet