CMSC733: Homework 0 - Alohomora

Kumar Gaurav
Department of Computer Science
University of Maryland, College Park
(used 3 late days)

***Abstract*—Homework 0 was a detailed study of constituents of edges, lines and boundary, which are due to texture, color and brightness. This helps in basic object recognition. Object recognition was done in classical method using pb-Lite approach and deep learning models. For deep learning approach CIFAR-10 dataset was used on CNN type of architecture. Results varied from 68 to 75 percent on test data.**

I. Phase 1

Phase 1 focuses on texture, color and brightness of image. Firstly, filter banks are applied to extract texture of an image. Three types of filter banks are used. Derivative of gaussian filter is kind of first smoothing the image and then differentiating it. Similarly, there is LM filter bank that differentiates, double differentiates and smoothed images at different scales and orientation for finding discontinuity in different orientations. Gabor filters are considered as one of the best filter for recognizing textures.

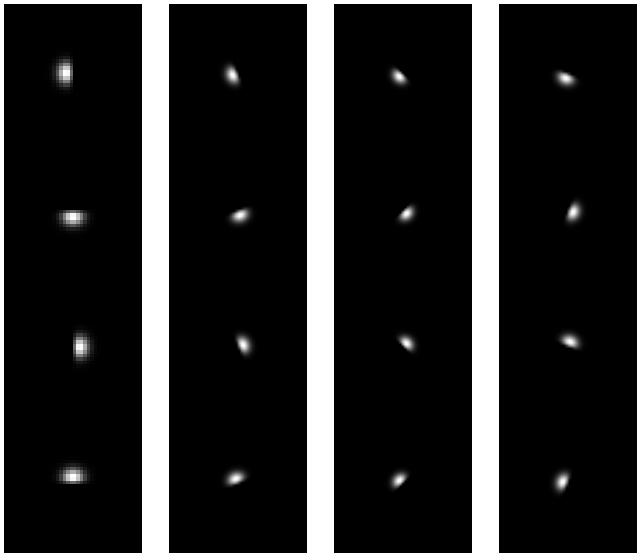

Fig. 1: DoG filter at scale=1.0

Second filter that was implemented was LM filter. This filter has set of first derivative filter which was at 3 scales and different orientations each. It also consisted of Gaussian filters at 4 scales and Laplacian filter which could be implemented by subtracting two gaussians at different scales. Set of second derivative filter can be seen in Fig3.

Thirdly Gabor filters are shown as the last set of filters from filter bank.

Below are the figures for Laplacian filters from LM filter bank at varios scales.

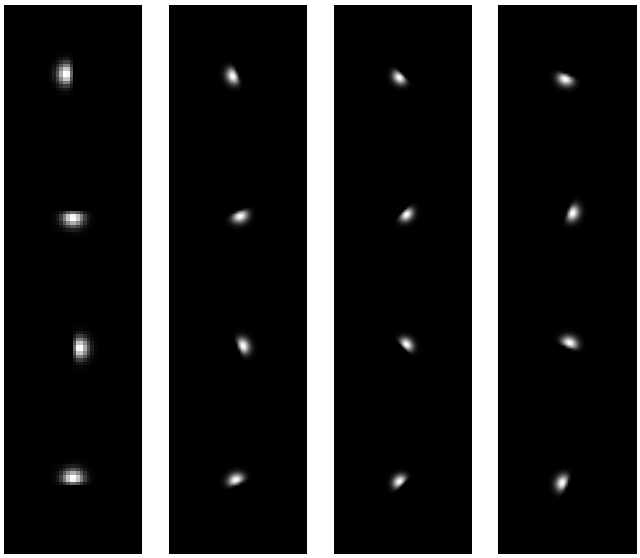

Fig. 2: DoG filter at scale=1.4

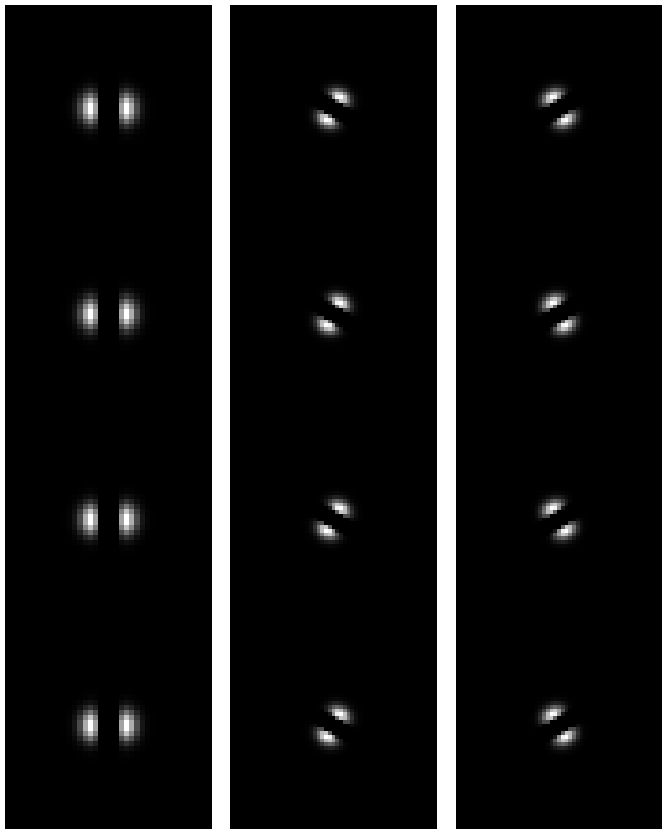

Fig. 3: LM second derivative filter at scale=1.0 and 1.4

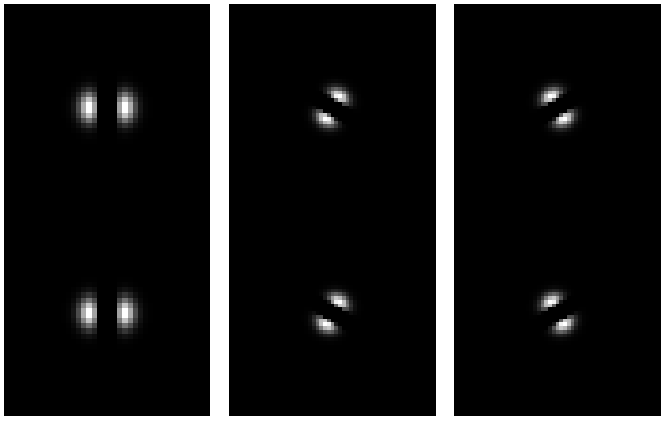

Fig. 4: LM second derivative filter at scale=2.0

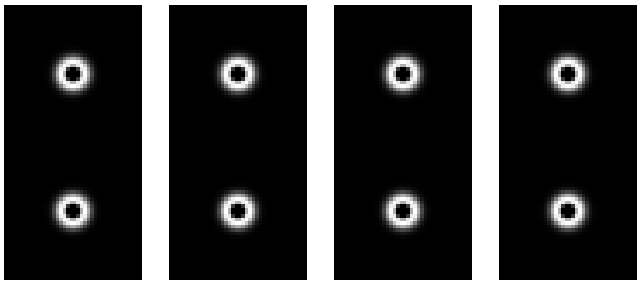

Fig. 5: LOG filters at various scales

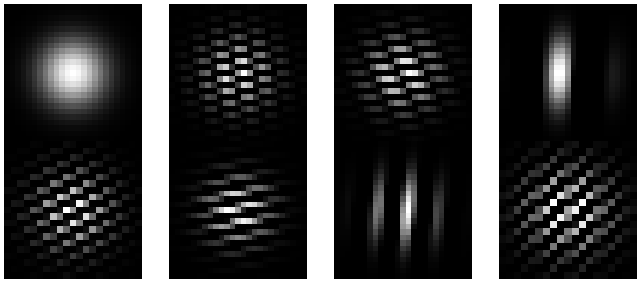

Fig. 6: LOG filters at various scales

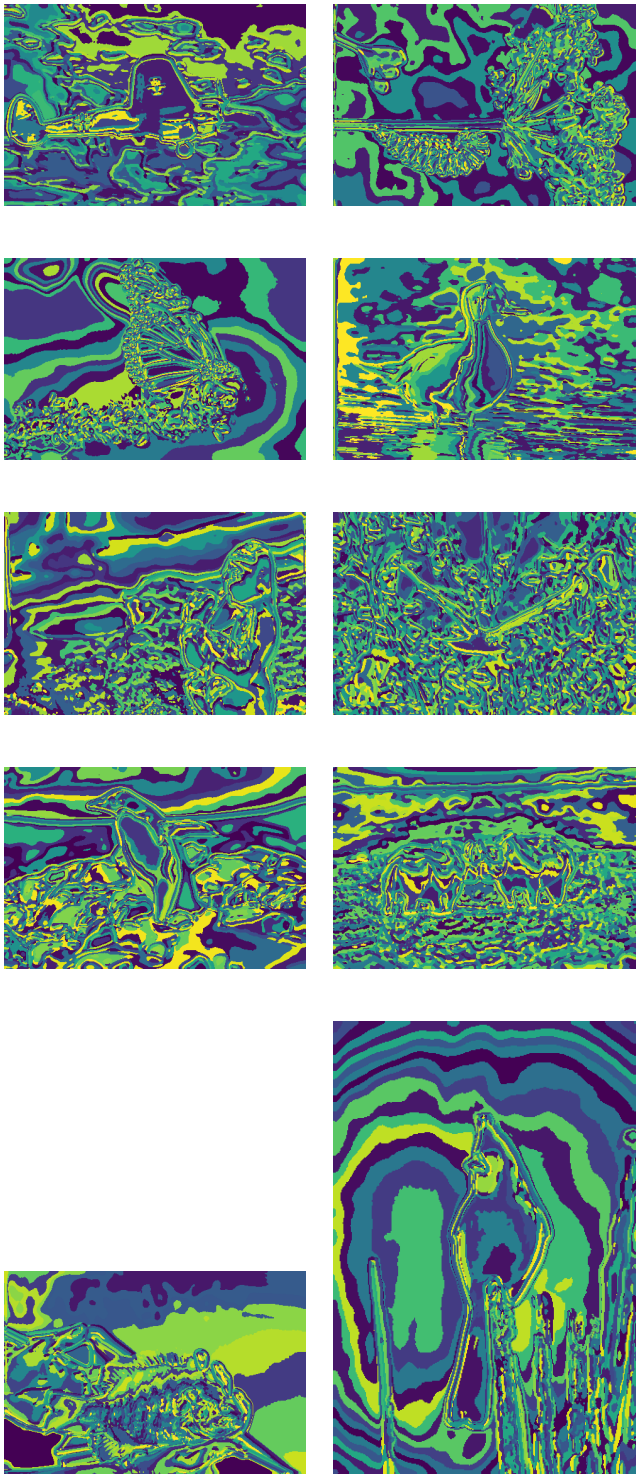

Fig. 7: Texton map of all images

After generating the filters, filters are applied to images each pixel, thus generating features that is textures. Mow, using kmeans these features are distributed in clusters and thus Texture map is formed. Following are the texture map of ten true images.

Gray scale images have brightness as a feature, In the similar manner as texton this feature is clustered using kmeans and Brightness map is created.

Next and last Color map is created by clustering RGB images using kmeans and assigning cluster value to each pixel.

Half mask disks are then created for gradient calculation and thus chi squared gradient calculation done to get gradient for texture, color and brightness.

Next and last Color map is created by clustering RGB images using kmeans and assigning cluster value to each pixel.

Final steps involve enhancing the features from canny and sobel images to get better edges using simple equation. The magnitude of the features represents the strength of boundaries, hence, a simple mean of the feature vector at location i should be somewhat proportional to pb.

Finally, the PBlite images are shown for each of the test cases. These images are better than Canny or Sobel since it takes into account texture, color and brightness feature, then it combines with canny and sobel baselines. Thus, getting true and enchanced edges of object.

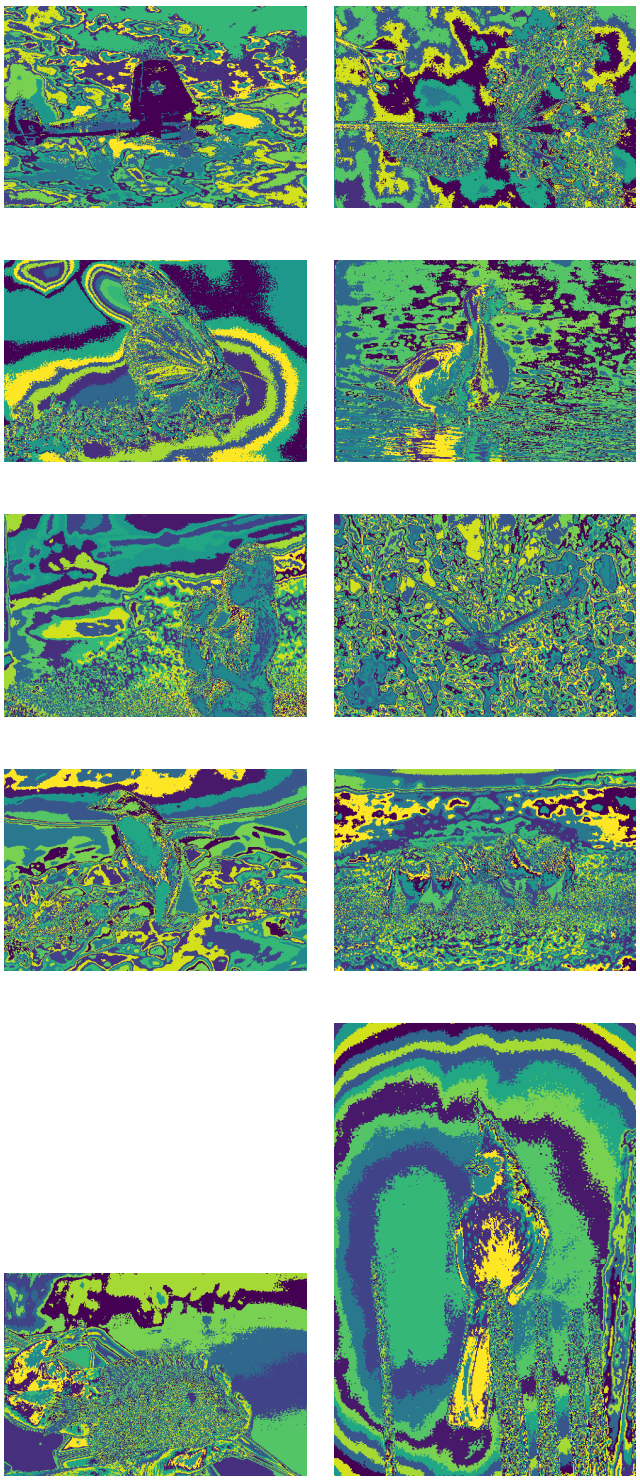

Fig. 8: Brighton map of all images

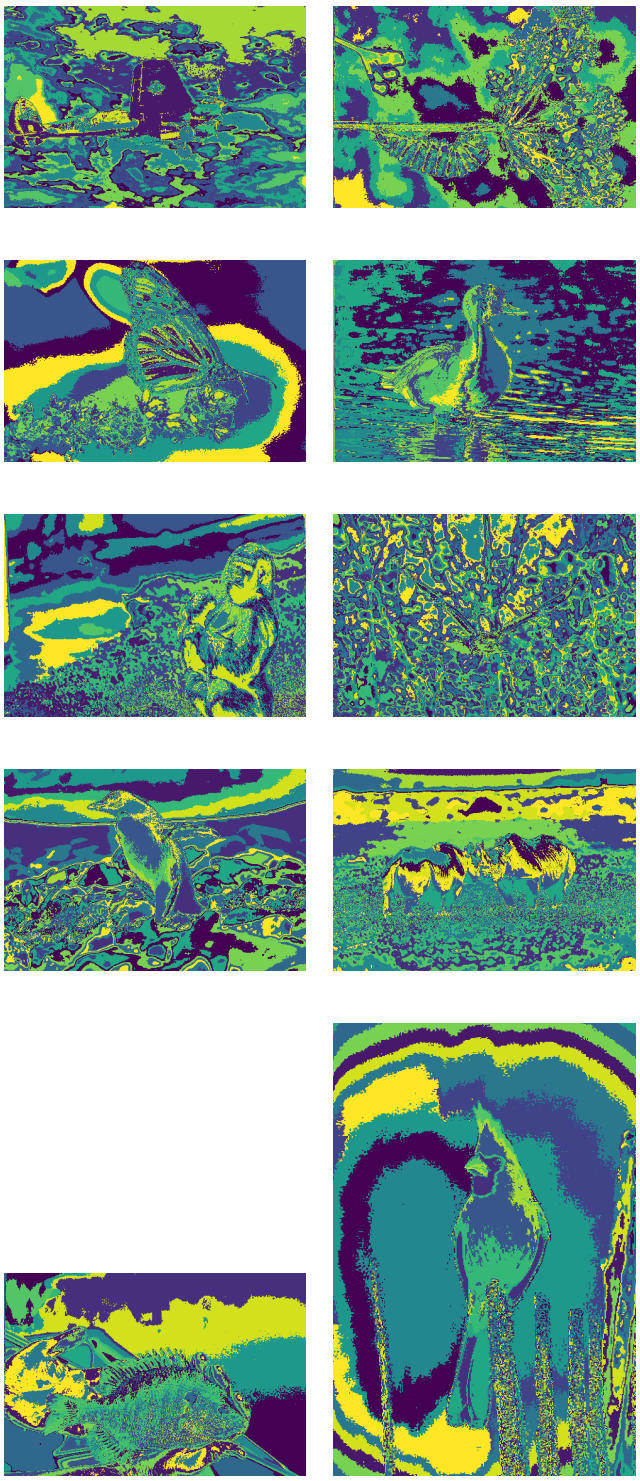

Fig. 9: Color map of all images

II. Deep Learning

First, I tried a simple CNN model. My machine is not equipped with GPU so, it was very slow. I ran the code on spyder so, no tensorboard model was given after compilation. My model was CONV32-RelU-CONV64-ReLu-CONV64-ReLu-FC128-FC10. Test accuracy was 67percent. While the train accuracy was 78 percent after 20 epochs. Keras was not at all used. Confusion matrix for testing data present below.

III. Modified model

For modified model I had used VGG Net which has deeper network and has 3 Convolutional layer. Each convolutional layer has two 2d convolution layer and maxpooling layer. I did also added dropout of 0.2 and choose kernel as uniform. Dropout actually makes the system ignore few neurons. Thus, train accuracy increased to 87 percent while test accuracy was 70 percent. Confusion matrix for testing data present below.

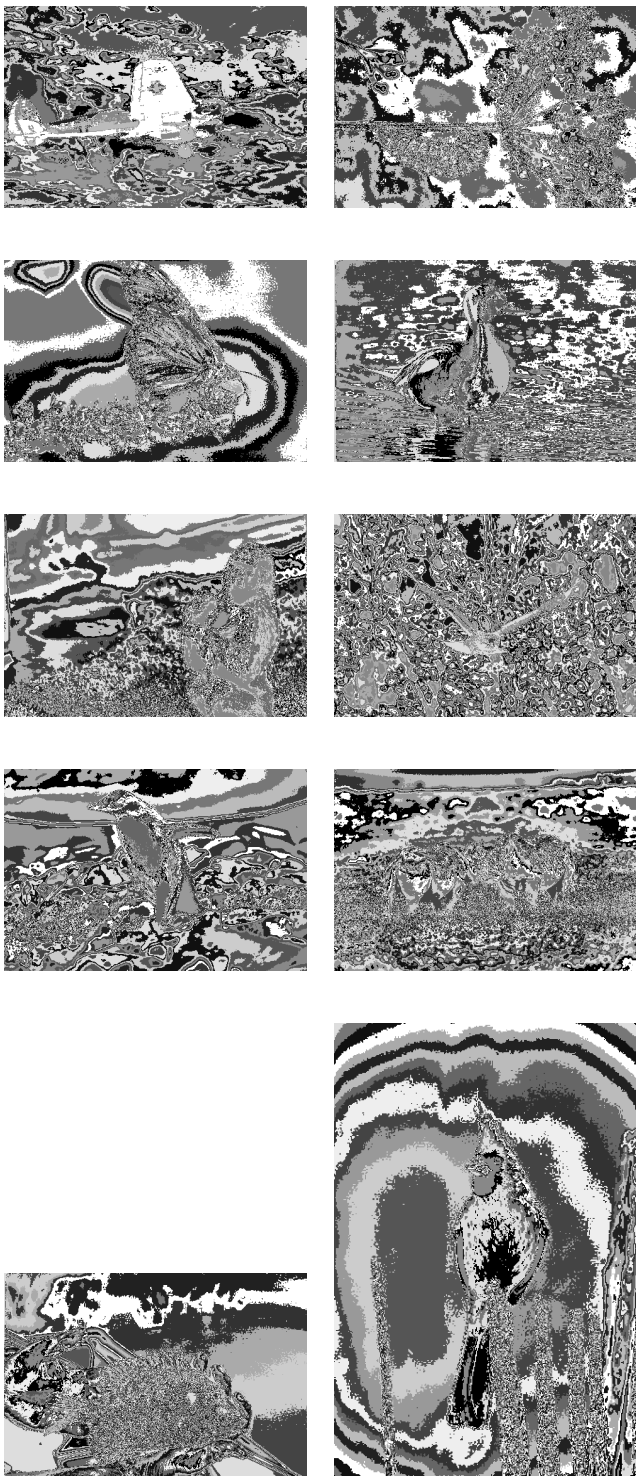

Fig. 10: Brightness gradient of all images

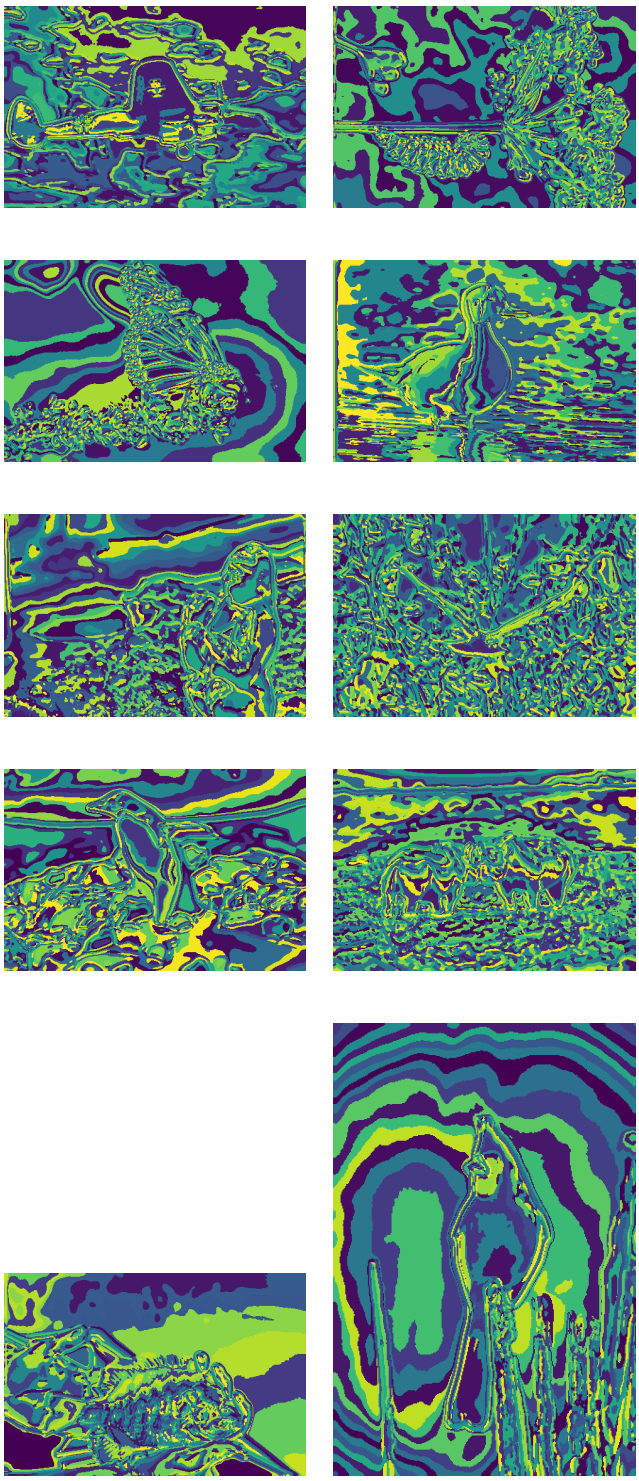

Fig. 11: Texture gradient of all images

IV. ResNet model

ResNet model architecture consited of five layer of convolution bolck and identity block. But my system was unable to train on that since it crashed in the midst of training several times. The architecture is present in Network.py.

V. Conclusion

Modified model performed better than simple model by increase of 3 percent in accuarcy in test data while I was unable to train any deep architectures due to limitations of my system.

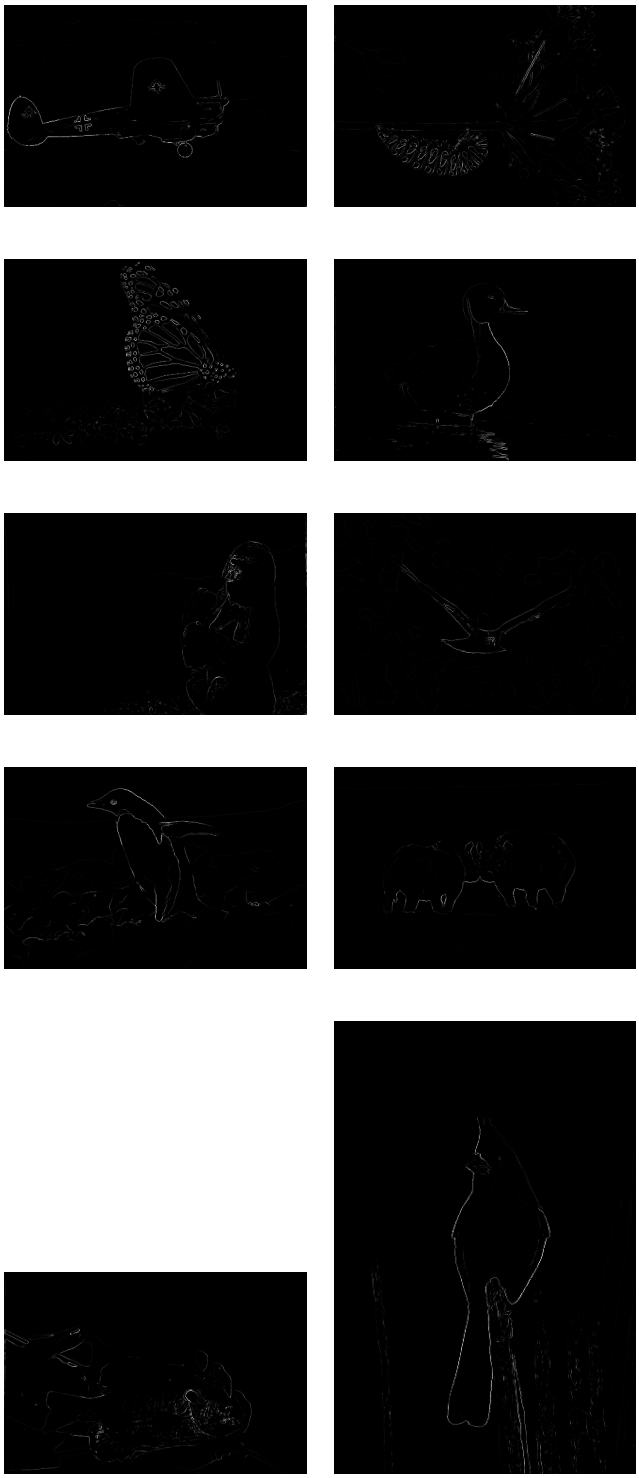

Fig. 12: Texture gradient of all images

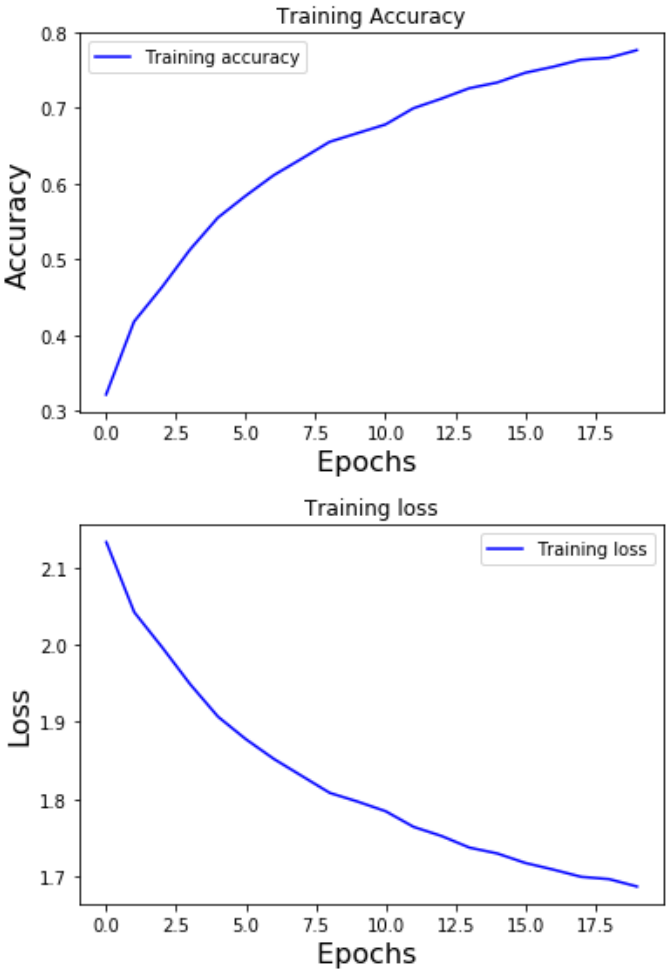

Fig. 13: Training Accuarcy and loss for simple model

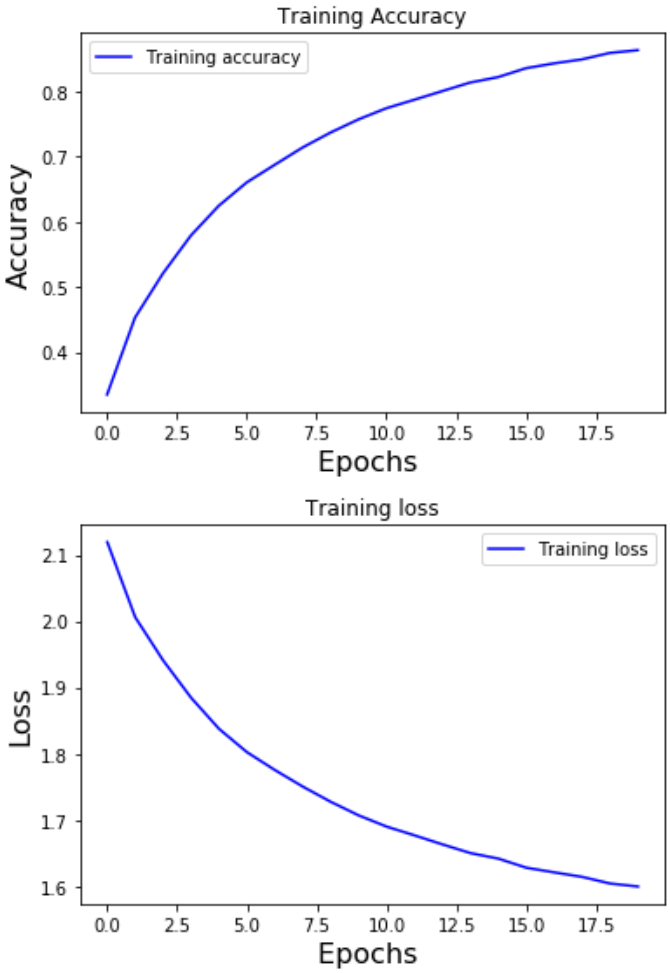

Fig. 14: Training Accuarcy and loss for modified model

```
<Figure size 432x288 with 0 Axes>
[786  33  30  28   5   3   6  21  56  32] (0)
[ 13 868   4  16   1   1  13   6  21  57] (1)
[ 87  11 524 121  49  79  40  69  14   6] (2)
[ 29  18  41 615  32 155  41  45  10  14] (3)
[ 43   9  83 133 487  53  53 126  11   2] (4)
[ 15   7  44 200  20 603  32  63   8   8] (5)
[ 10   8  38 106  26  40 754   5   8   5] (6)
[ 16   7  32  51  25  53   2 803   3   8] (7)
[ 72  66   9  23   5   6   6  10 776  27] (8)
[ 46 158   8  44   1   8  18  21  24 672] (9)
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
('Accuracy: 68.88', '%')
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
[768  24  28  15   9   2  11  11 101  31] (0)
[ 17 844   6  10   5   3   8   2  42  63] (1)
[ 87   7 563  62  93  62  65  29  24   8] (2)
[ 37  20  70 474  71 138 109  22  38  21] (3)
[ 32   9  88  44 663  27  64  48  18   7] (4)
[ 22   9  56 144  61 588  49  44  19   8] (5)
[ 10  10  39  33  35  17 829   4  21   2] (6)
[ 38  11  33  39  59  69  16 709  12  14] (7)
[ 36  34   9   3   7   5   6   3 886  11] (8)
[ 35 121   6  16  10   7  13  12  39 741] (9)
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
('Accuracy: 70.65', '%')
```

Fig. 15: Confusion matrix of testing data
above:Simple model, below:Modified MOdel