Report
HW0

Gnyana Teja Samudrala
UID:115824737

I. Phase-1

The boundary detection in the single image is a difficult task. One can closely get the edges from the filters like Sobel and Canny. But they are not continous boundaries they get blended into the surroundings or even donot detect if its weak. So we make use of a method proposed as probability of boundary detection method to find the probability of a boundary to be present. A simpler version of this is implemented in this report, which is known as Pb-lite boundary detection. This makes use of the filter banks, textons, brightness and color map, also the knowledge from Sobel and Canny filters.

A. Methodology

The dataset of the images is provided and also their Sobel and Canny filtered images are also provided. So the implementation requires us to find the Texton Map which can be obtained by filtering the image through various kinds of filter banks.

$FilterBanks$

These are the collection of normal kernels of a specific or mixture of filters with different scales and orientations.

Derivative of Gaussian: The kernel of this filter is obtained by taking the centre pixel as origin and calculates the other corresponding values from the first derivative of the gaussian equation. To get this into implementation separated the 2D filter into two filters of X and Y componenets, which are represented by g_x and g_y. The equations of which are as follows

$$g_x = \frac{-x}{2\pi\sigma^4} e^{\frac{-(x^2+y^2)}{2\sigma^2}}, g_y = \frac{-y}{2\pi\sigma^4} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

These values are combined with the combination of angle values to get the desired rotated DoG kernal. The final equation would be

$$g = cos(\alpha)g_x + sin(\alpha)g_y$$

Leug-Malik: This filter bank is the combination of the first two derivatives of the Gaussian, Gaussian itself and Laplacian of Gaussian at different scales and orientations. There are two version of this filter bank short and long respectively known as LMS and LML.

The LMS consists of first and second derivatives of Gaussian at scales $\sigma = \{1, \sqrt{2}, 2\}$ with elogation factor of 3 (which means $\sigma_x = \sigma$ and $\sigma_y = 3*\sigma_x$. The next are Gaussians at four basic scales ($\sigma = \{1, \sqrt{2}, 2, 2\sqrt{2}\}$) and eight LoG filters with elongation factor 3. The long version occurs at these basic scales $\sigma = \{\sqrt{2}, 2, 2\sqrt{2}, 4\}$ The equations used for Laplacian of Gaussian is

$$LoG = \frac{x^2+y^2-2\sigma^2}{\sigma^4} * \frac{e^{\frac{-(x^2+y^2)}{2\sigma^2}}}{2\pi\sigma^2}$$

Gabor: The Gabor filter contains Gaussian kernal with a sinusoidal signal. So the equation would become complex. But to build the kernal only the real part of the equation is considered. So the equation forms up to be

$$g = e^{\frac{-(x_p^2+\gamma^2 y_p^2)}{2\sigma^2}} cos(2\pi\frac{x_p}{\lambda} + \psi)$$

$HalfDiskMasks$

The half disks masks are created using the DoG filters thresholded at specific values of low and high to get two masks making a pair of half disks.

$TextonMap$

The texton map is created by applying the oriented Gaussian filter bank and then K-Means clustering is done to get an ID for each cluster. The chi-square distance measure is calculated between the applied half disk mask pairs and a gradient value is obtained for the textons.

Similar to this the brightness (gray scale image) and the color gradient maps are created which are represented as B_g and C_g. All these gradients are combined in a ratio with the Sobel and Canny baseline images from which we get the probabilities of the boundaries detected. The equation of the probability is

$$P = \frac{T_g + B_g + C_g}{3}(0.5 * Sobel + 0.5 * Canny)$$

B. Results

The filter banks of all the three kinds are visualized in the images from figures:s 1- 3
The table gives the idea about the parameters used in building these kernals.

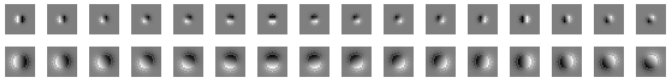

Fig. 1. Derivative of Gaussian filter bank

The next step is finding the texton map using the DoG filter bank of kernal size 49, and scale [1,2,4,8] 16 orientations in range [0,360]. From this we get an output of 64 channels and

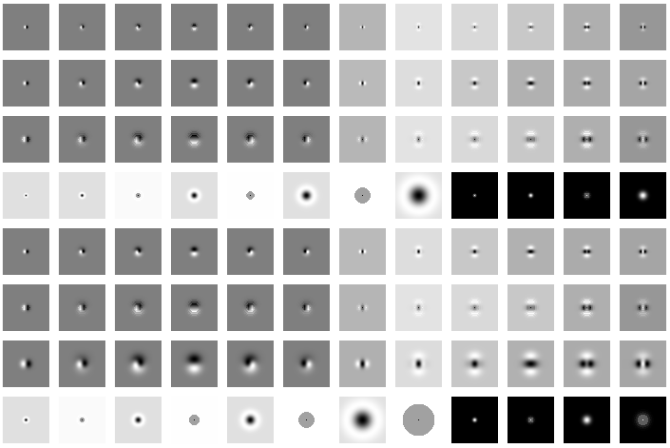

Fig. 2. Leug-Malik filter bank

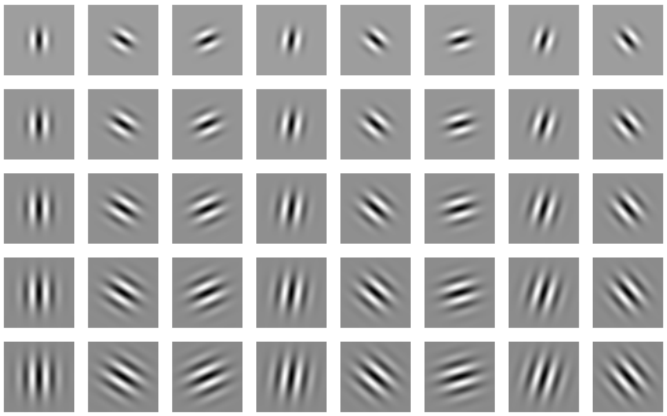

Fig. 3. Gabor filter bank

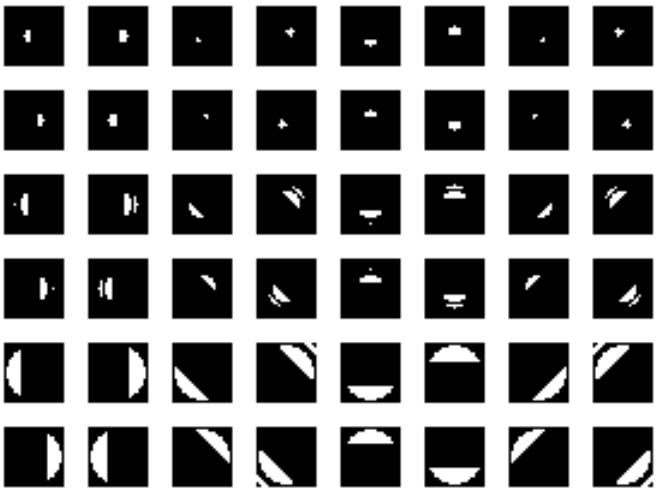

Fig. 4. Half disk masks

each pixel has 64 values which are clustered using K-Means. The output of the Texture map is as shown in fig. 5

The brightness and color gradients are as in figs. 6 and 7

TABLE I
FILTER BANK PARAMETERS

Filter Bank	Size	Scale	Orientation
DoG	25	2,4	16 equally in $[0^0, 240^0]$
LMS	49	$1, \sqrt{2}, 2, 2\sqrt{2}$	6 equally in $[0^0, 180^0]$
LML	49	$\sqrt{2}, 2, 2\sqrt{2}, 4$	6 equally in $[0^0, 180^0]$
Gabor	49	[5,10]	8 equally in $[0^0, 360^0]$

Fig. 5. Texton Map of 1st image

Fig. 6. Brightness Map of 1st image

So the final boundary probability is in fig. 8

II. PHASE - 2

In this to find the class the image belongs to is implemented by Convolution Neural Networks using Tensorflow. The first architecture of the network is built with the help of the starter code and the tutorials given. Then a few tweaks are performed on that network like the filter sizes, strides, layers etc.,

First Neural Network

The architecture of the network is as given in the graph of the tensor board fig. 18

Fig. 7. Color Map of 1st image

Fig. 8. Pb-lite boundary of 1st image

Fig. 9. Pb-lite boundary of 2nd image

The parameters of the network used to get the presented results are in the table II. This is a very basic type of network.

Fig. 10. Pb-lite boundary of 3rd image

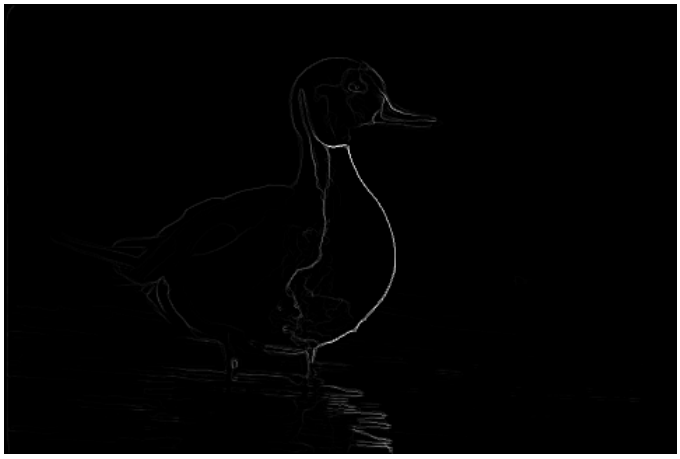

Fig. 11. Pb-lite boundary of 4th image

Fig. 12. Pb-lite boundary of 5th image

Second Network

In this network the architecture is changed by adding few more layers to the network and changing the size of layers.Batch normalization is performed in every layer, so the

TABLE II
THE PARAMETERS USED IN NEURAL NETWORKS IMPLEMENTED

Parameter	First	Second	Third(ResNet)
Optimizer	Adam	Momentum	Momentum
Learning Rate	1e-4	10e-3	1e-2
Momentum	N/A	0.9	0.9
Batch Size	1	100(+5 every 10 epochs)	120(+10 every 5 epochs)
Epochs	35	40	80
Data augmentation	Nill	flipping	flipping
Time of run	7:12:27	00:41:00	01:25:59
Number of parameter	1,429,322	1,437,834	570,970
Accuracy on test images	72.12	78.53	79.2
Accuracy on train images	97.716	93.824	89.278

TABLE III
THE ARCHITECTURE OF NEURAL NETWORKS IMPLEMENTED

Stage	First	Second	Third(ResNet)
Conv1	[5,5,3,64]	[5,5,3,64]	[3,3,3,16]
Pool1	Max.pool(stride=2)	Max.pool(stride=2)	Avg.pool(stride=2)
Conv2	[5,5,64,64]	[5,5,64,64]	5times of [[3,3,16],[3,3,16]]
Pool2	Max.pool(stride=2)	Max.pool(stride=2)	N/A
Conv3	[5,5,64,128]	[5,5,64,128]	5times of [[3,3,32],[3,3,32]]
Pool3	Max.pool(stride=2)	Max.pool(stride=2)	N/A
Conv4	[5,5,128,256]	[5,5,128,256]	5times of [[3,3,64],[3,3,64]]
Pool4	Max.pool(stride=2)	Max.pool(stride=2)	Global average
local3	flatten(256)	flatten(256)	flatten
local4	128	128 ,64	N/A
Softmax	10	10	10

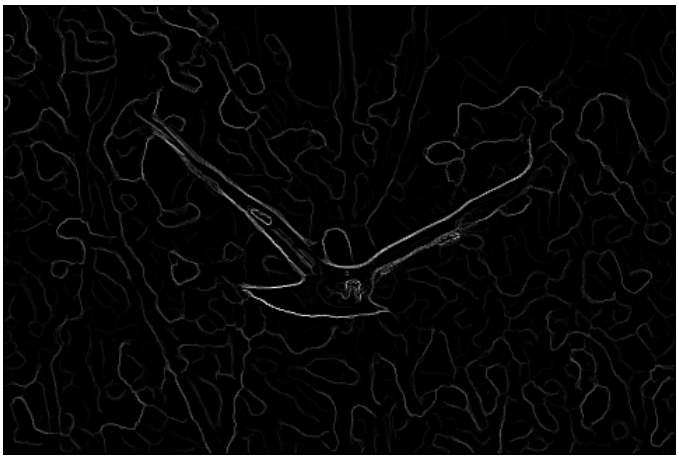

Fig. 13. Pb-lite boundary of 6th image

Fig. 14. Pb-lite boundary of 7th image

bias layers were no longer needed. The image before passing to the neural network it is standardized i.e the range of values are changed from [0,255] to [-1,1]. Also the images are flipped to increase the number of training data.

$$Increasing \quad the \quad batch \quad size$$

Instead of decaying the learning rate the batch size is increased. The learning rate is kept constant and the batch size is increased at every 10 epochs interval by a value of 5. The optimizer used is the Momentum optimizer with a momentum of 0.9. The learning rate is increased compared to the previous network as the batch size is increased the learning pace should also be increased proportionally.

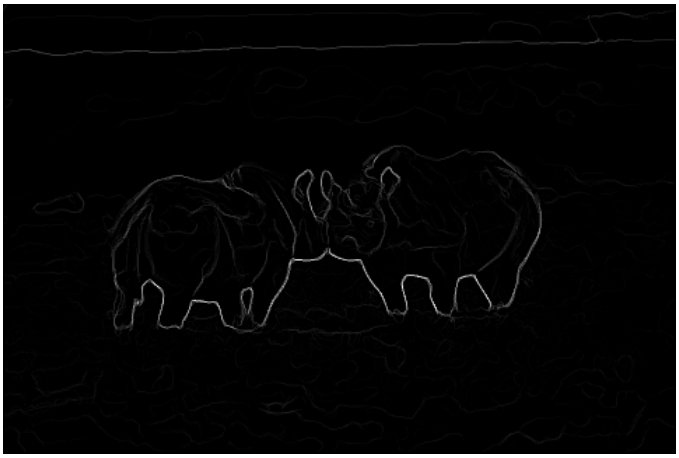

Fig. 15. Pb-lite boundary of 8th image

Fig. 16. Pb-lite boundary of 9th image

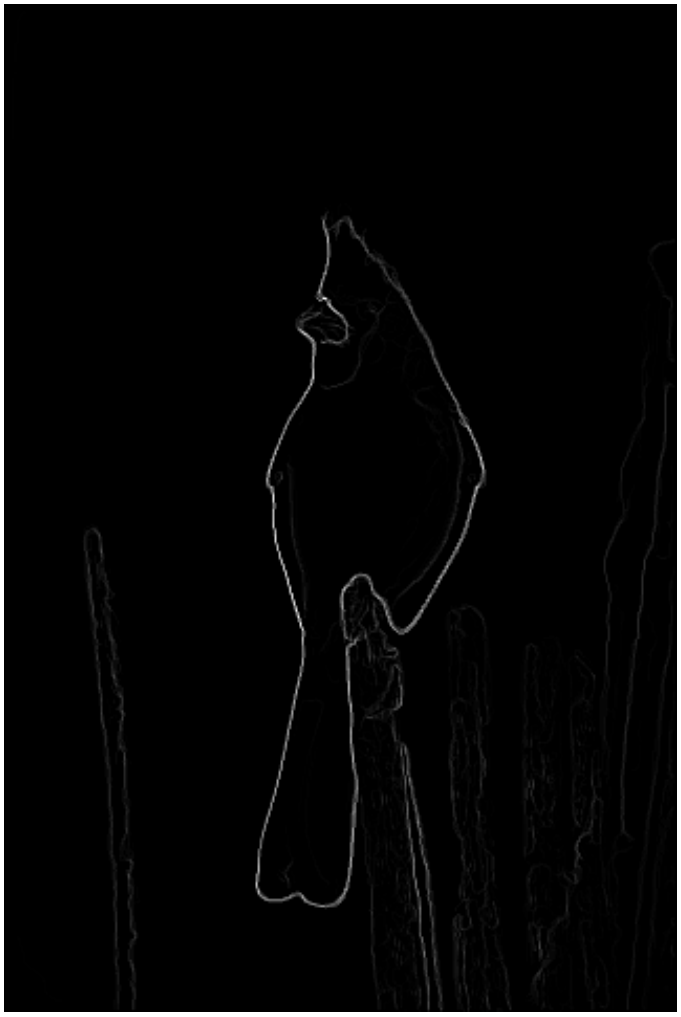

Fig. 17. Pb-lite boundary of 10th image

By making all these changes to the network the performance of the network increased and the accuracy also increased by a great factor.

ResNet

The architecture implemented is as layed out in the ref paper. The ResNet-32 architecture is choosen and it has 16,32,64 number of filters and each block has five convolutional layers. Each block of convolutional layer has a shortcut path connected. To match the sizes the shortcut signal is strided and padded. The filter sizes are all of 3X3. This network is quick to achieve desired accuracy compared to the rest of the networks.

The convolution blocks consists of two convolution layers each of size 3X3, the first convolution layer has a filter which is convolved with the input and then passed through batch normalization and relu activation. The second layer doesnot have relu activation but after it gets added to the shortcut the activation is performed. At the starting of each block the size of output is reduced by using a stride of 2 in the convolution layer.

Results

The accuracy over each epoch is carried over the test and train data and plotted in the graphs, the first 1000 images are taken for this purpose. From the table II we can see the accuracy increases as the convolution layers increases, most importantly the accuracy on the test set increases which is required. The parameters in the resnet is also less and the time taken is also less for the resnet to be trained, as the epochs for which it ran is greater.

REFERENCES

[1] DONT DECAY THE LEARNING RATE,INCREASE THE BATCH SIZE $https://arxiv.org/pdf/1711.00489.pdf$

[2] Deep Residual Learning for Image Recognition $https://arxiv.org/pdf/1512.03385.pdf$

[3] $http://6.869.csail.mit.edu/fa16/lecture/lecture3linearfilters.pdf$

[4] $http://matlabserver.cs.rug.nl/edgedetectionweb/web/edgedetection params.html$

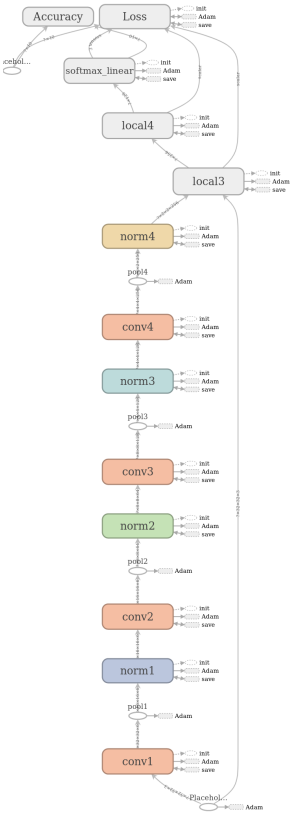

Fig. 18. Graph of the first network

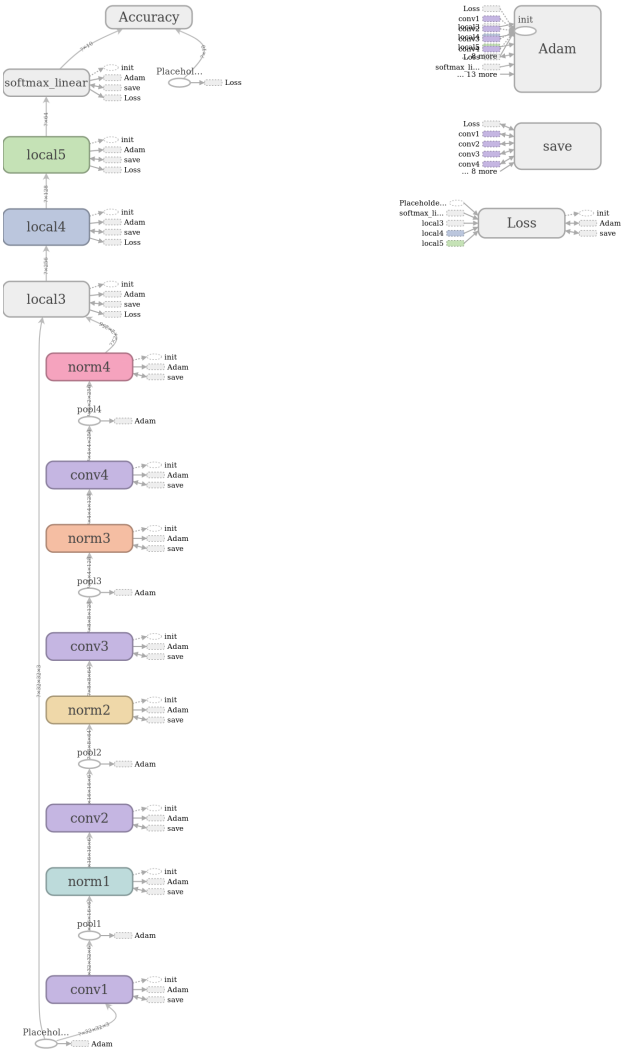

Fig. 19. Graph of the second network

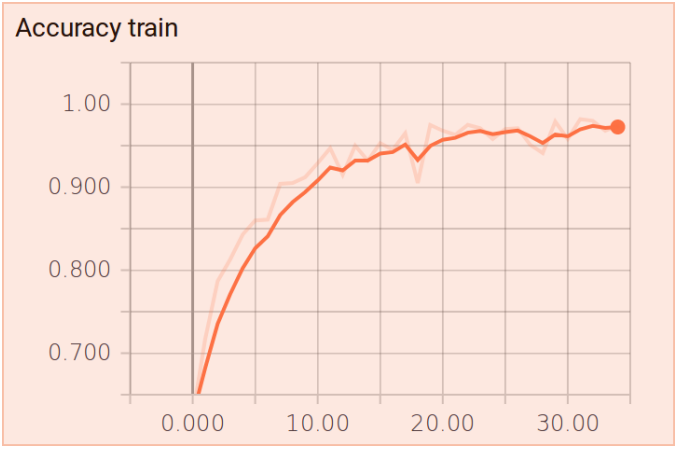

Fig. 20. Accuracy over each epoch on train data of 1st network

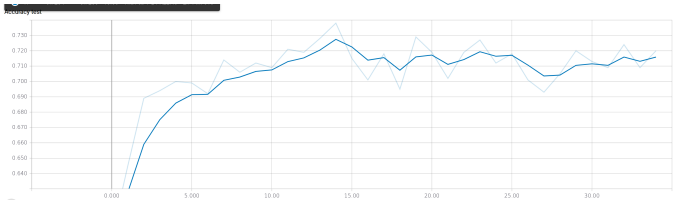

Fig. 21. Accuracy over each epoch on test data of 1st network

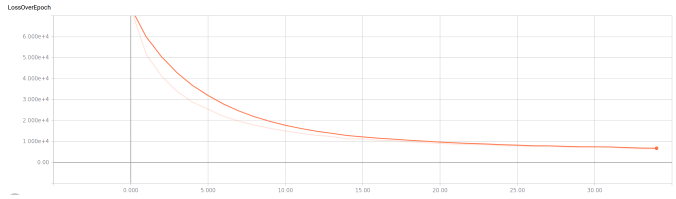

Fig. 22. Loss over epoch of 1st network

```
Number of parameters in this model are 1429322
100%|                                        | 50000/50000 [02:36<00:00, 318.92it/s]
[4920    2   37    2    6    0    0    6    9   18] (0)
[   1 4975    1    0    0    3    6    0    3   11] (1)
[  11    1 4957    2    6   12    4    4    1    2] (2)
[  14    1  102 4680   10  129   22   29    9    4] (3)
[  20    0   92    9 4773   22   14   67    1    2] (4)
[   0    1   44   25    7 4885    6   28    1    3] (5)
[   1    3   43   11    8   23 4898    8    4    1] (6)
[   5    0    5    1    2   11    2 4966    1    7] (7)
[  56   19   10    9    3    0    7    0 4891    5] (8)
[   3   57    5    3    0    3    1    5   10 4913] (9)
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
('Accuracy: 97.716', '%')
```

Fig. 23. Confusion matrix on training set of 1st network

```
Number of parameters in this model are 1429322
100%|                                        | 10000/10000 [00:32<00:00, 309.91it/s]
[772  19  80  16  17   7   9   9  44  27] (0)
[  7 864   7  12   4   8   9   7  21  61] (1)
[ 56   2 717  39  48  52  40  33   7   6] (2)
[ 26  11 114 457  48 184  59  64  10  27] (3)
[ 34   4 114  49 595  53  44  88   8  11] (4)
[ 18   1  84 143  29 624  29  52   9  11] (5)
[  5   5  65  45  32  37 781   8  11  11] (6)
[ 18   1  49  24  31  53   6 801   4  13] (7)
[ 75  33  21  15   5  10   8   5 813  15] (8)
[ 27  84  16  13   4  21   7  17  23 788] (9)
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
('Accuracy: 72.12', '%')
```

Fig. 24. Confusion matrix on test set of 1st network

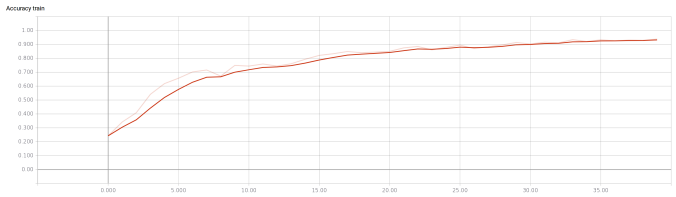

Fig. 25. Accuracy over each epoch on train data of 2nd network

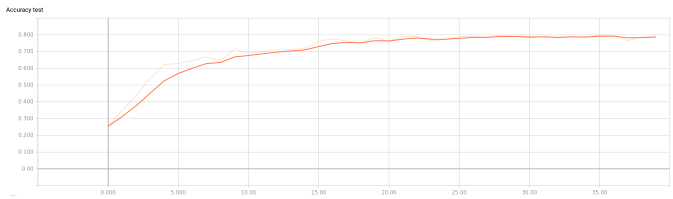

Fig. 26. Accuracy over each epoch on test data of 2nd network

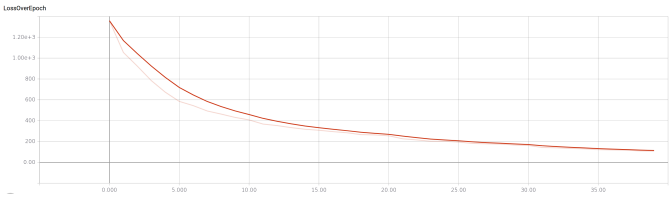

Fig. 27. Loss over epoch of 2nd network

```
Number of parameters in this model are 1437834
100%|                                        | 50000/50000 [02:38<00:00, 311.03it/s]
[4805   20   27   15   10    2    9   34   45   33] (0)
[   3 4962    3    3    1    2    9    1    3   13] (1)
[  99    6 4616   38   52   62   98   25    4    0] (2)
[  25    5   74 3837   57  817   77   95    5    8] (3)
[  10    1   62   24 4619  103   46  126    8    1] (4)
[   1    1   32   85   14 4774   12   78    3    0] (5)
[   7    4   30   40    9   36 4869    2    0    3] (6)
[   7    0   15   20   16   82    5 4855    0    0] (7)
[  20   51   18    6    0    6   16    3 4870   10] (8)
[  12  224    2    7    0    4    6   27   13 4705] (9)
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
('Accuracy: 93.824', '%')
```

Fig. 28. Confusion matrix on training set of 2nd network

```
Number of parameters in this model are 1437834
100%|                                        | 10000/10000 [00:32<00:00, 309.43it/s]
[838  10  36   9   9   2   9  16  47  24] (0)
[  3 938   3   7   1   5   5   2   8  28] (1)
[ 68   6 684  30  48  48  78  29   6   3] (2)
[ 15   4  70 465  38 278  56  55   8  11] (3)
[  9   0  64  33 710  58  47  72   5   2] (4)
[  6   2  31  89  23 788  18  39   3   1] (5)
[  6   5  26  34  18  28 873   5   3   2] (6)
[ 10   0  15  21  25  61   3 862   1   2] (7)
[ 37  36  13   7   5   4  11   2 866  19] (8)
[ 20 100   8   7   0   3   5  15  13 829] (9)
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
('Accuracy: 78.53', '%')
```

Fig. 29. Confusion matrix on test set of 2nd network

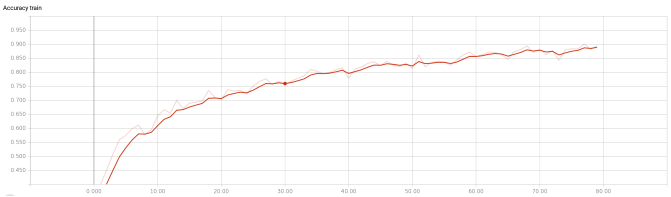

Fig. 30. Accuracy over each epoch on train data of 3rd network

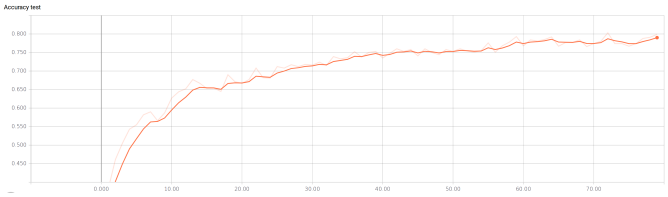

Fig. 31. Accuracy over each epoch on test data of 3rd network

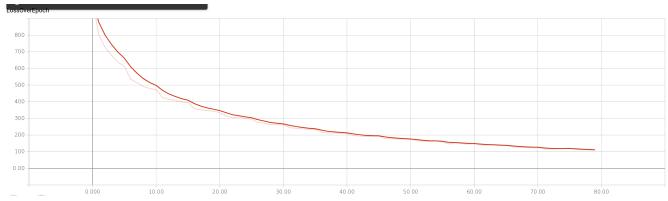

Fig. 32. Loss over epoch of 3rd network

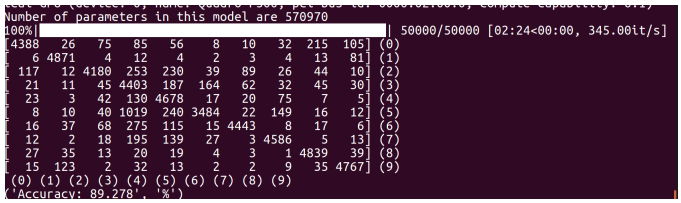

Fig. 33. Confusion matrix on training set of 3rd network

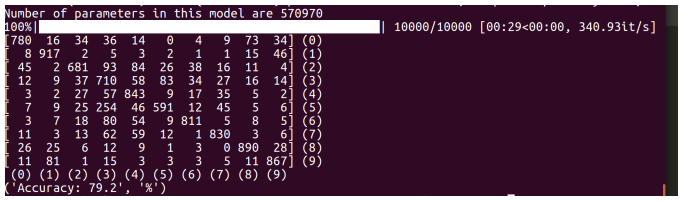

Fig. 34. Confusion matrix on test set of 3rd network