Homework 0: Alohomora

Gowri Shankar Sai Manikandan [1]
Using 1 Late Day

1. Introduction

Phase 1: Shake My Boundary:

In Phase 1, the aim was to implement pb-lite boundary detection, using several mathematical based filter banks. The final intended result was to get the edges in an images. The major steps involved in this segment of the assignment are- Filter Bank Creation, Generation of Texture, Color, and Brightness Maps, and then using Half Disk Masks, their gradients as well. Then finally, boundary detection was carried out.

2. Filter Banks

2.1. Oriented DoG Filter

The first filter bank implemented was the Oriented Derivative of Gaussian (DoG) filter bank. It was generated by convolving sobel filter with Gaussian kernel and then rotating for a given angle. Two scales, 2 and 1 were considered, along with 16 different orientations from 0 to 360 degrees. Figure 1 represents the DoG filter bank.

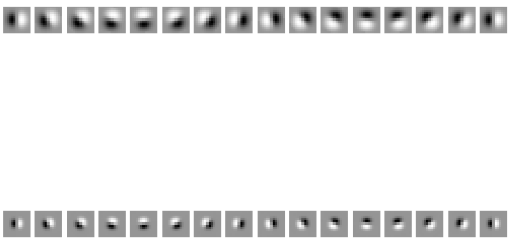

Fig. 1. Oriented Derivative of Gaussian (DoG) filter bank

2.2. Leung Malik Filters

Two Leung Malik filter bank were generated, namely, Leung Malik Small (Figure 2), and Leung Malik Large (Figure 3). It contains First and Second Order Derivatives of Gaussians, at 6 different orientations and 3 scales, along with Laplace of Gaussian Filters, and Gaussian Filters.

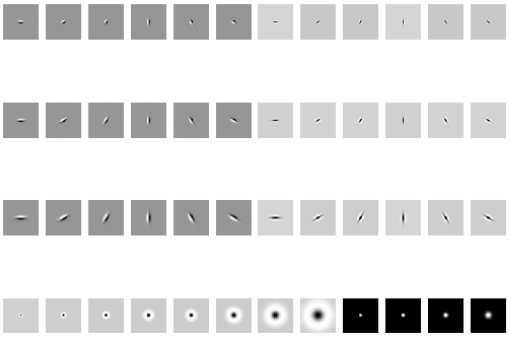

Fig. 2. Leung Malik Small Filter Bank

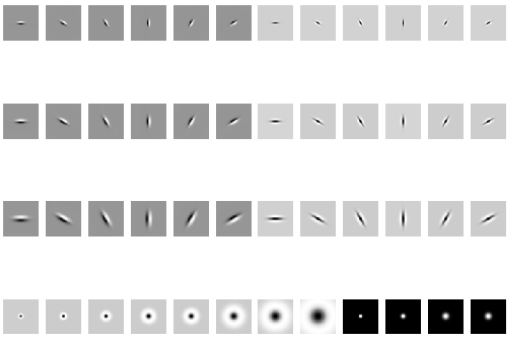

Fig. 3. Leung Malik Large Filter Bank

[1]Worcester Polytechnic Institute. Correspondence to: Gowri Shankar Sai Manikandan <gmanikandan@wpi.edu>.

RBE/CS 549 Computer Vision, Worcester Polytechnic Institute.

2.3. Gabor Filters

The last filter bank generated was the Gabor filter bank (Figure 4). It was built on 5 scales, and 8 orientations. A gabor filter is mathematically defined as a sinusoidal wave, multiplied by a gaussian function.

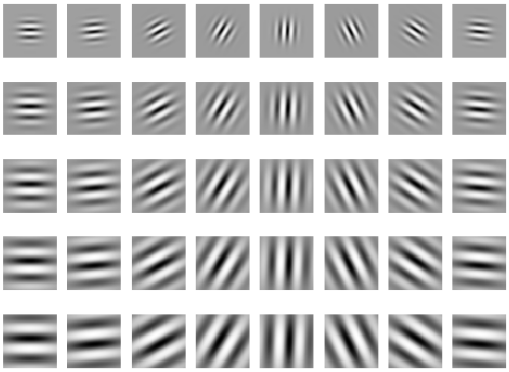

Fig. 4. Gabor Bank

3. Texton, Brightness and Color Maps

Next step in the process is to create Texton, Brightness and Color Maps. In brief, a texton map involves filtering the image through all the filter banks generated. Then, using KMeans clustering, we can receive a texton id for each pixel, which previously had number of values equal to filters. Similar steps can be followed to receive the brightness and color maps, for which greyscale image and RGB image were used respectively. Given below are the maps for all images.

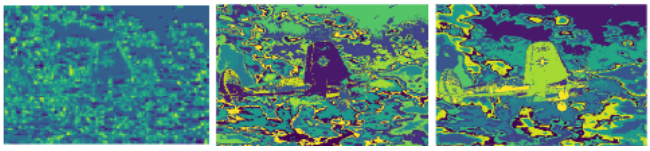

Fig. 5. Texton, Brightness and Color Maps of Image 1

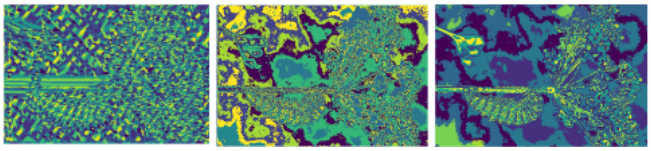

Fig. 6. Texton, Brightness and Color Maps of Image 2

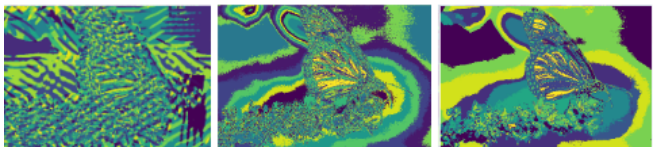

Fig. 7. Texton, Brightness and Color Maps of Image 3

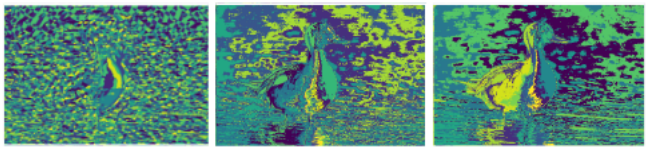

Fig. 8. Texton, Brightness and Color Maps of Image 4

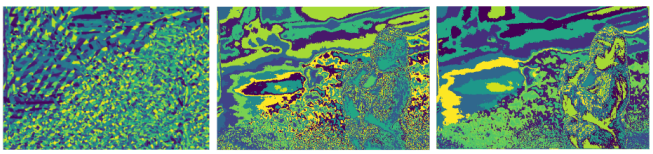

Fig. 9. Texton, Brightness and Color Maps of Image 5

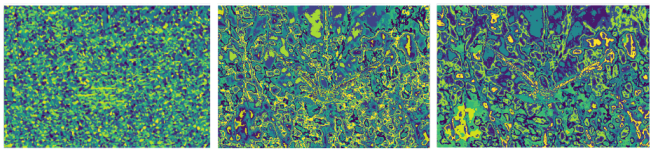

Fig. 10. Texton, Brightness and Color Maps of Image 6

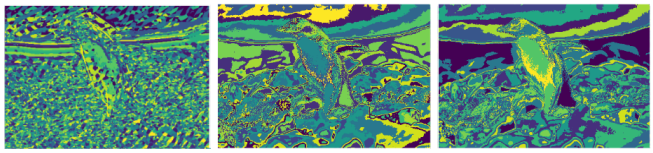

Fig. 11. Texton, Brightness and Color Maps of Image 7

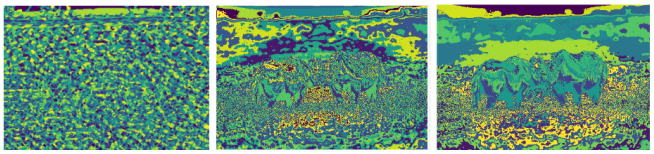

Fig. 12. Texton, Brightness and Color Maps of Image 8

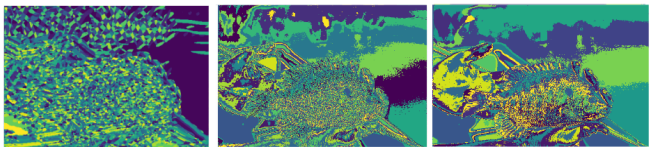

Fig. 13. Texton, Brightness and Color Maps of Image 9

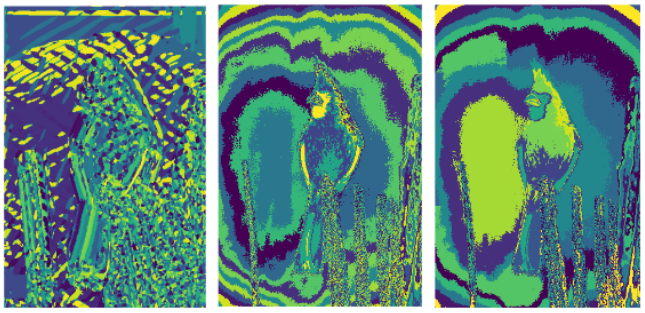

Fig. 14. Texton, Brightness and Color Maps of Image 10

4. Half Disk Mask

Half Disk Mask (Figure 15) were created, with alternating left and right pairs,in opposite orientations for 8 different

orientations, and 3 different scales, 5,10 and 15.

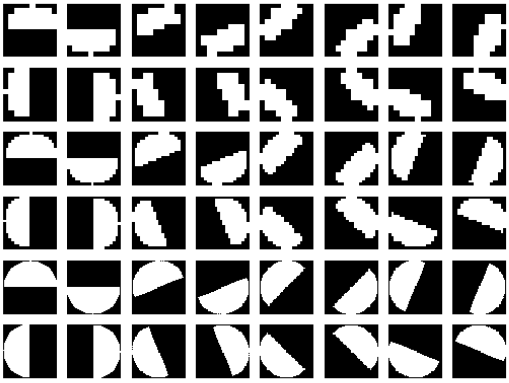

Fig. 15. Half Disk Masks

5. Texture, Brightness and Color Gradients

Using the above generated half disk masks, and chi-square distance, texture, brightness and color gradients were produced. Given below are the gradients for all images.

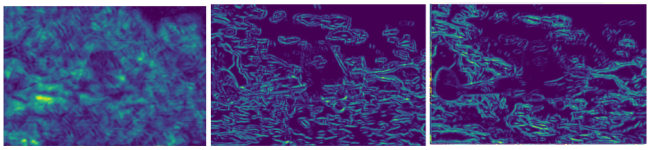

Fig. 16. Texton, Brightness and Color Gradients of 1

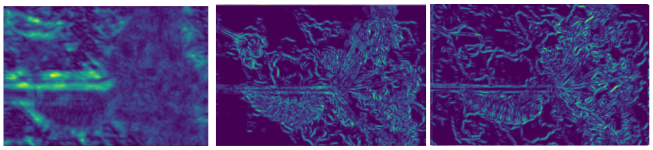

Fig. 17. Texton, Brightness and Color Gradients of 2

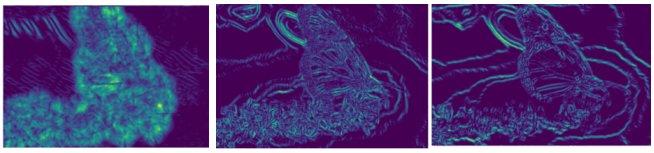

Fig. 18. Texton, Brightness and Color Gradients of 3

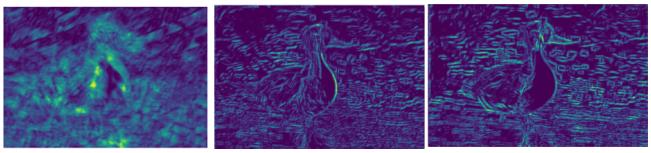

Fig. 19. Texton, Brightness and Color Gradients of 4

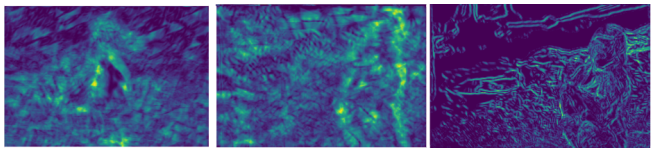

Fig. 20. Texton, Brightness and Color Gradients of 5

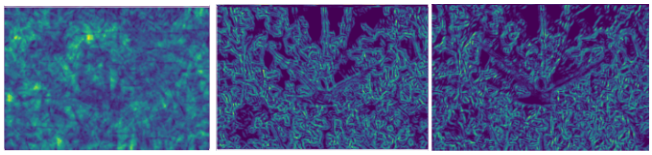

Fig. 21. Texton, Brightness and Color Gradients of 6

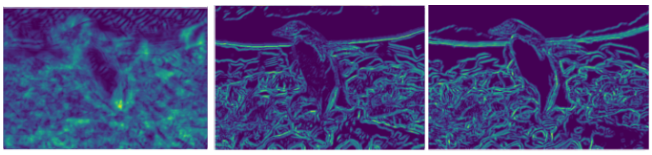

Fig. 22. Texton, Brightness and Color Gradients of 7

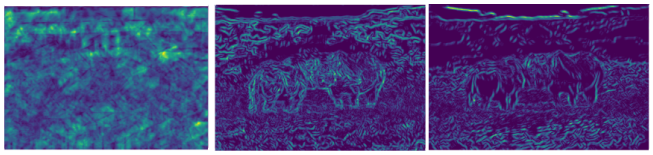

Fig. 23. Texton, Brightness and Color Gradients of 8

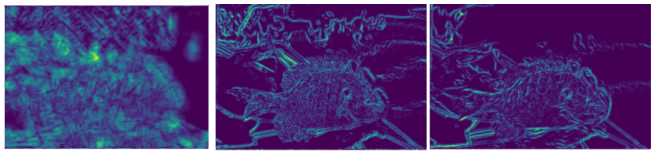

Fig. 24. Texton, Brightness and Color Gradients of 9

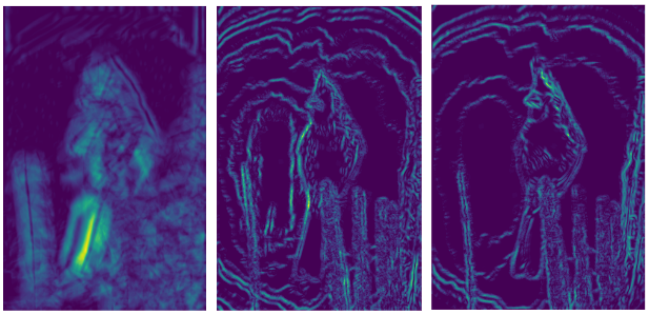

Fig. 25. Texton, Brightness and Color Gradients of 10

6. Pb-lite Output

In the end, using hadamard operator, we combine the sobel and canny baselines, along with the generated gradients. Following is a comparison between the Canny, Sobel and Pb-lite output.

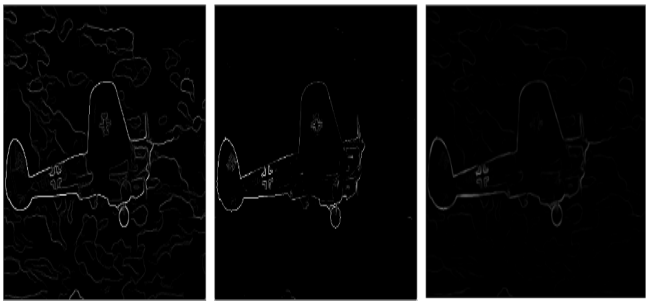

Fig. 26. Canny, Sobel and Pb-lite Output of 1

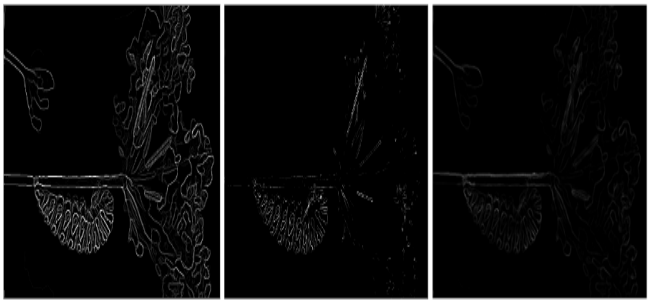

Fig. 27. Canny, Sobel and Pb-lite Output of 2

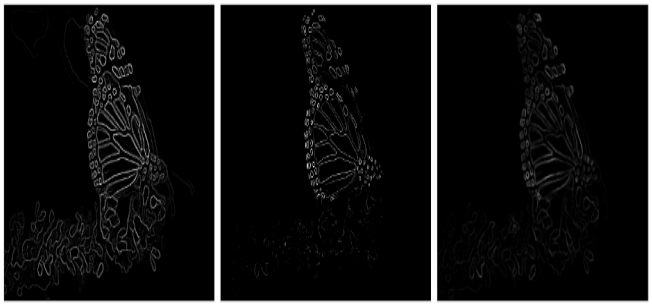

Fig. 28. Canny, Sobel and Pb-lite Output of 3

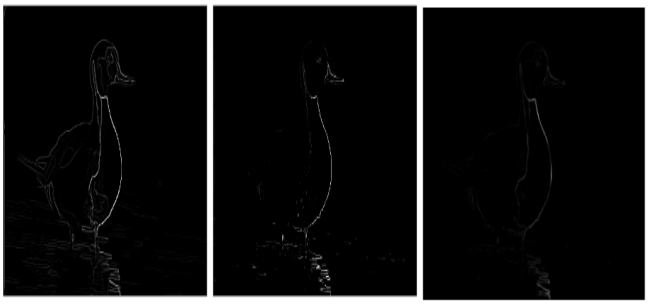

Fig. 29. Canny, Sobel and Pb-lite Output of 4

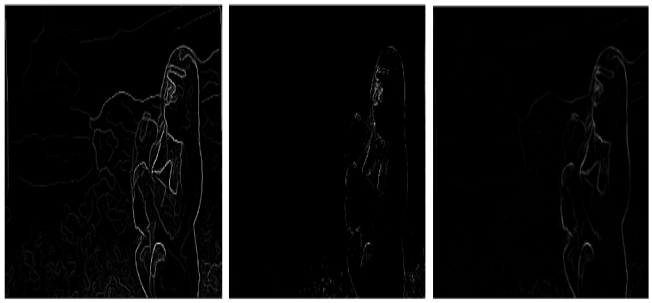

Fig. 30. Canny, Sobel and Pb-lite Output of 5

Fig. 31. Canny, Sobel and Pb-lite Output of 6

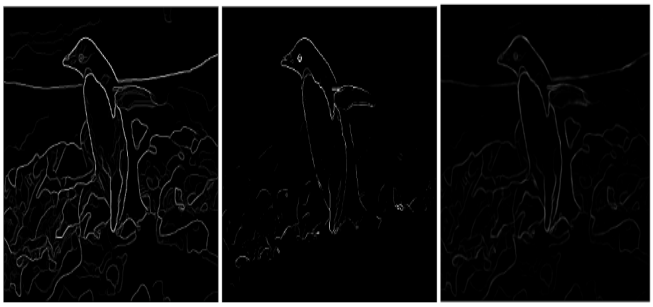

Fig. 32. Canny, Sobel and Pb-lite Output of 7

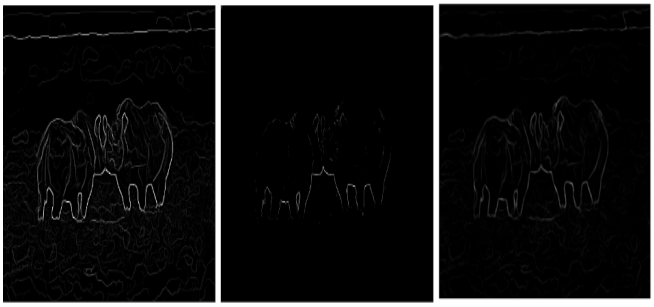

Fig. 33. Canny, Sobel and Pb-lite Output of 8

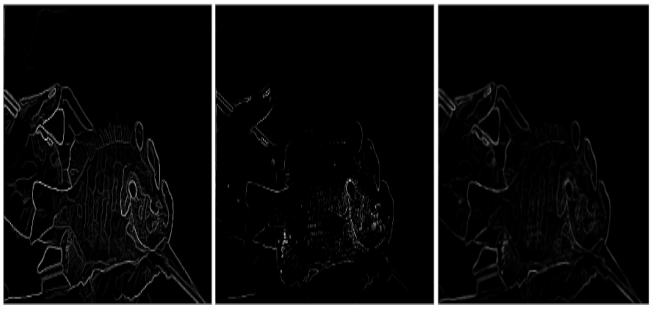

Fig. 34. Canny, Sobel and Pb-lite Output of 9

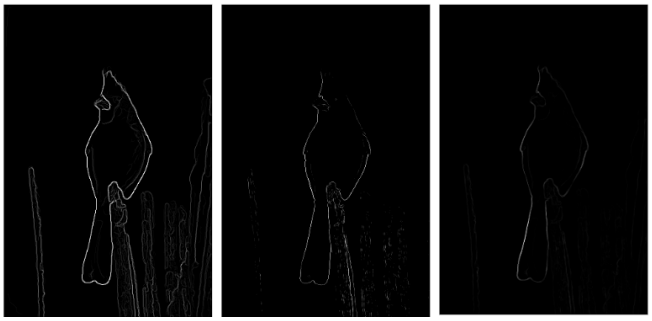

Fig. 35. Canny, Sobel and Pb-lite Output of 10

7. Analysis

From what I inferred from the results of Pb-lite is that unlike Canny, which considers unwanted edges, and Sobel, which in some cases tend to miss out on important boundaries, Pb-lite gives the appropriate edges that are required for detection. Also, changing the weights for Canny and Sobel baselines can give a significant change in Pb-lite's output, which is a useful technique for optimization of Pb-lite output. Also, I think parameters of the filter banks can be tuned to get better pb-lite output.

Phase 2: Deep Dive on Deep Learning:

8. First Neural Network

The architecture of my first neural network is in Fig. 36. 15 Epochs were taken, with 25 Mini Batch size. I opted for Adam optimizer in this, with learning rate as 1e-3. In summary, the model has two convolutional layers, two max pool layers following the convolutional layers, and Relu activation functions, along with dropout layer. Contrary to frequent use of Relu after convolution layer, I tried Relu activation after MaxPool layers. The total number of parameters in this model were 21,482.

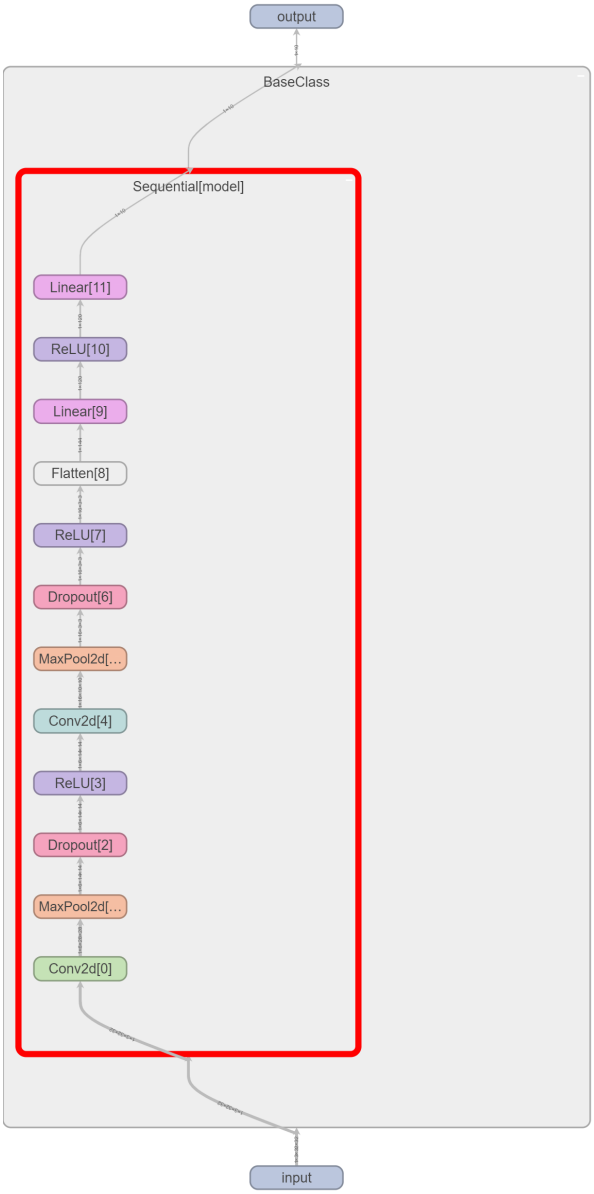

Fig. 36. First Neural Network

I	0	1	2	3	4	5	6	7	8	9
0	3042	233	260	68	123	18	121	156	761	218
1	137	3873	53	23	27	9	83	76	289	430
2	334	95	2281	295	670	103	669	274	193	86
3	86	107	360	1817	458	418	969	459	207	119
4	182	67	356	213	2768	73	582	579	122	58
5	39	91	344	1121	449	1561	541	620	124	110
6	35	73	215	228	329	20	3909	72	57	62
7	59	92	186	212	399	134	155	3571	70	122
8	342	203	45	60	57	13	81	32	3955	212
9	171	923	52	73	47	21	127	177	355	3054

Table 1. Confusion Matrix of First Model on Training set (Accuracy - 59.622 Percent)

Table 1. shows that the confusion matrix of this model on

training set. 59.22 percent accuracy was achieved on training, while testing set gave an accuracy of 55.56. Table 2. gives the confusion matrix for the same. Next, Fig. 37-39 shows the loss and accuracy over epochs, on training set, along with test set for loss.

I	0	1	2	3	4	5	6	7	8	9
0	570	51	72	11	28	8	29	23	160	48
1	41	721	12	8	8	1	26	15	68	100
2	68	17	405	76	136	30	152	62	36	18
3	22	20	69	333	98	81	191	100	52	34
4	22	21	78	42	502	22	136	131	32	14
5	13	12	79	222	97	280	105	139	31	22
6	5	10	38	55	74	6	775	11	12	14
7	24	19	43	40	89	38	35	663	10	39
8	83	54	22	10	18	5	14	13	738	43
9	36	207	14	8	10	8	24	35	89	569

Table 2. Confusion Matrix of First Model on Testing set (Accuracy - 55.56 Percent)

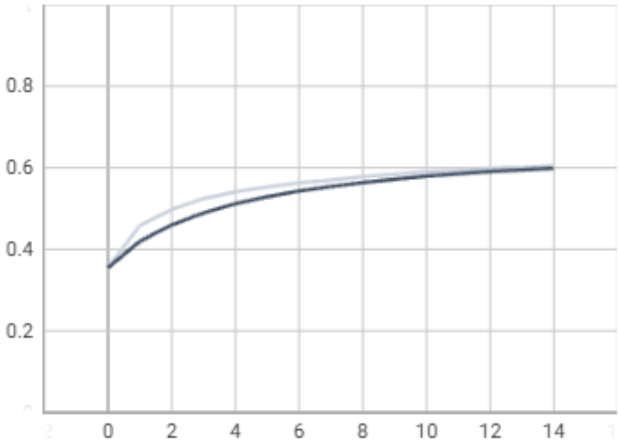

Fig. 37. Accuracy over Epochs on train set of First Model.

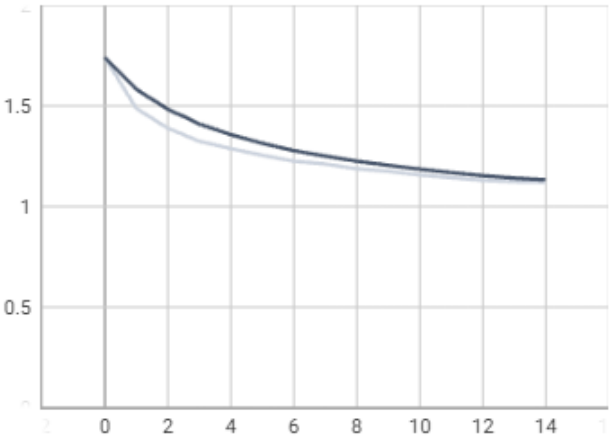

Fig. 38. Loss over Epochs on train set of First Model.

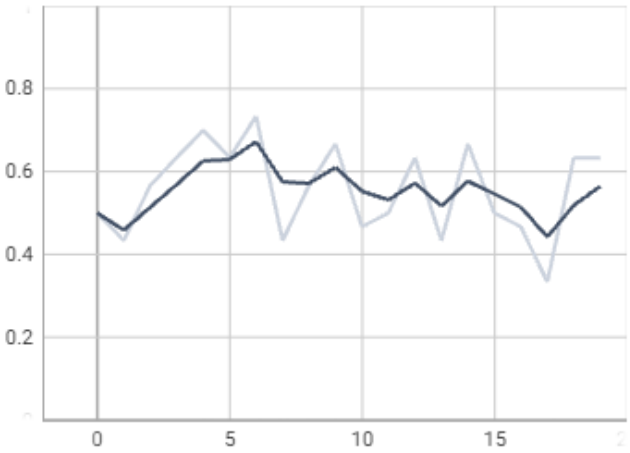

Fig. 39. Accuracy over Epochs on Test Set of First Model.

9. Improved Neural Network

In order to increase the accuracy on train set, the number of epochs were increased to 20, along with increase in mini batch size to 30. Additionally, batch normalization layers have been addded after every convolution layer. Subsequently, the train set accuracy increased to 63.642 percent, while the test set accuracy increased to 58.41 percent. Adam optimizer was used, with 1e-3 learning rate. Figure 40 shows the improved architecture. The number of parameters in the model was 21,526. Figure 41-43 shows the accuracy over test and train set, and loss over train set, while Table 3 and Table 4 present the confusion matrix for the train and test sets.

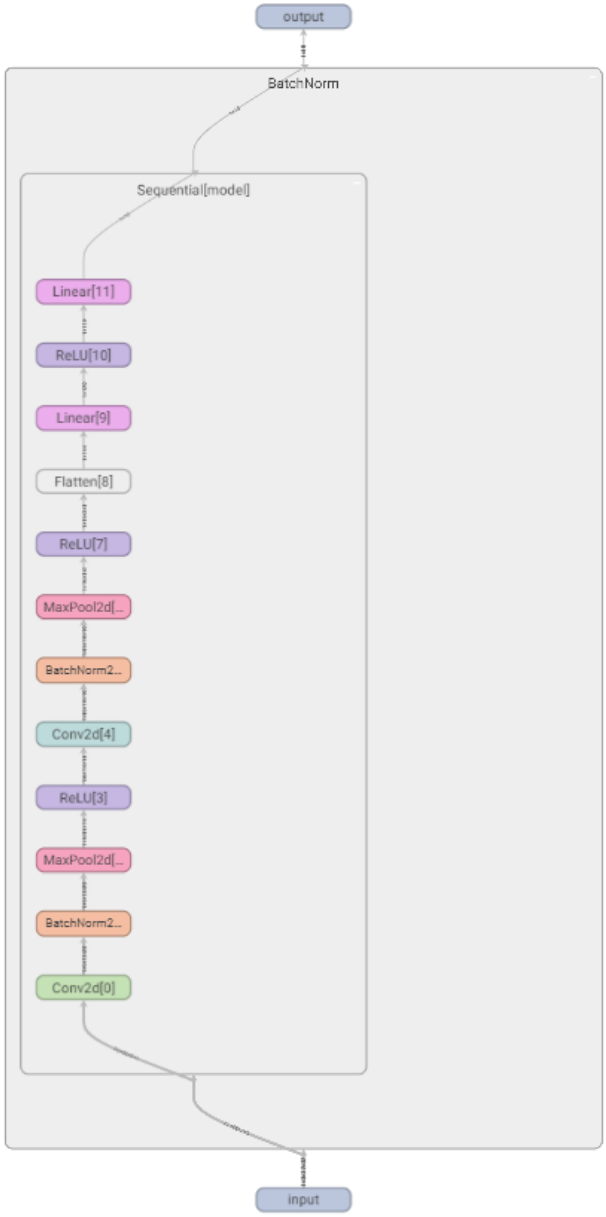

Fig. 40. Improved Neural Network

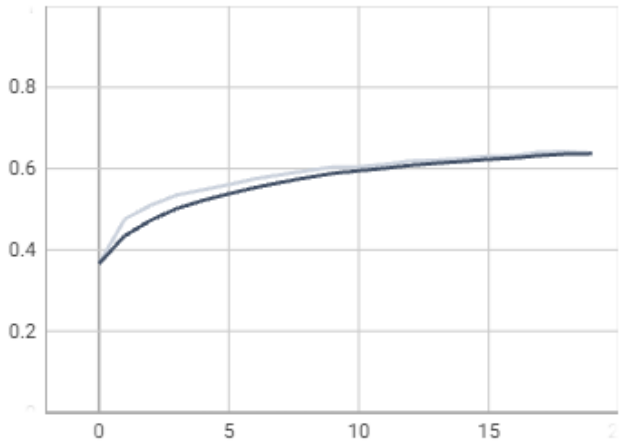

Fig. 41. Accuracy over Epochs on train set of Improved Model.

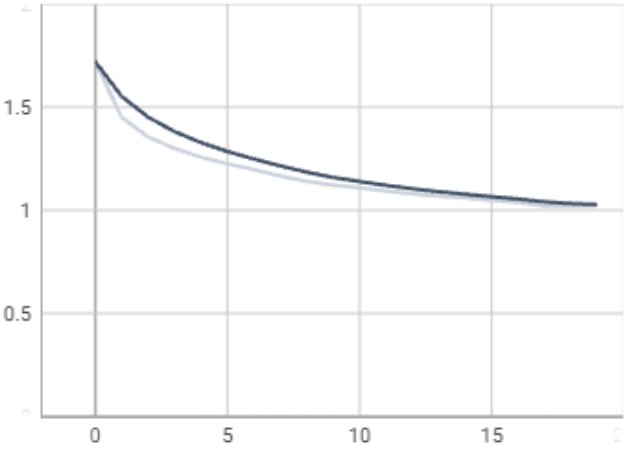

Fig. 42. Loss over Epochs on train set of Improved Model.

1	0	1	2	3	4	5	6	7	8	9
0	3543	204	256	119	70	63	55	35	447	208
1	166	3949	49	78	22	27	56	17	166	470
2	386	65	2680	460	370	383	405	83	90	78
3	130	62	401	2615	147	929	446	71	87	112
4	209	38	548	484	2539	342	418	275	61	86
5	55	40	299	1157	175	2788	238	126	39	83
6	60	66	300	452	190	145	3670	24	36	57
7	142	50	248	416	345	566	110	2909	20	194
8	494	214	113	127	33	53	30	9	3723	204
9	219	749	67	148	23	84	65	37	203	3405

Table 3. Confusion Matrix of Improved Model on Testing set (Accuracy - 63.642 Percent)

1	0	1	2	3	4	5	6	7	8	9
0	496	30	119	24	46	7	45	16	157	60
1	19	693	14	23	5	5	38	7	33	163
2	39	10	481	60	109	77	146	44	16	18
3	7	10	91	344	74	201	180	53	15	25
4	10	6	102	55	539	43	108	113	14	10
5	14	4	80	169	63	493	85	71	7	14
6	3	8	46	54	41	18	806	12	3	9
7	18	9	33	48	93	107	47	606	7	32
8	60	34	30	34	25	4	30	6	719	58
9	27	100	24	23	7	18	38	38	61	664

Table 4. Confusion Matrix of Improved Model on Testing set (Accuracy - 58.41 Percent)

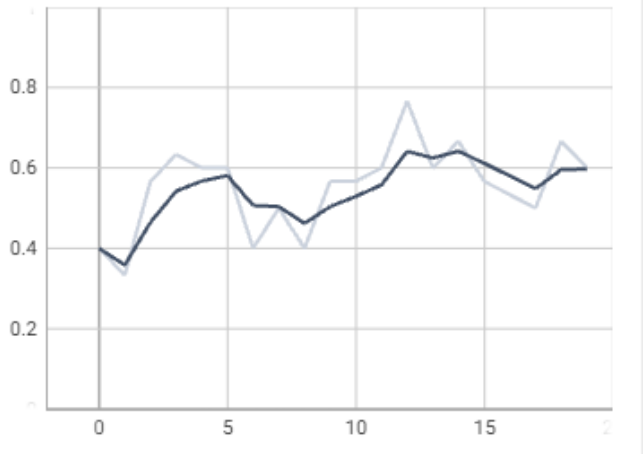

Fig. 43. Accuracy over Epochs on Test Set of Improved Model.

10. ResNet

The ResNet architecture implemented is given in Figure. 45. Initially, the input passes through a sequence of Convolution, Batch Norm, Relu activation, and Max Pooling. Then comes the main block, which is repeated four times. This block is the building block of ResNet architecture. Figure. 44 explains it better. So, first a 1*1 convolution takes place, followed by a 3 * 3 and then a 1 * 1 convolution. These convolution layers are followed by BatchNorm and Relu layers as well. The resulting output is then added with the origin input of the block, and then passed through Relu as input for the next block. In Figure 45, the four-blocks contain 3, 4, 6 and 3 such blocks respectively.

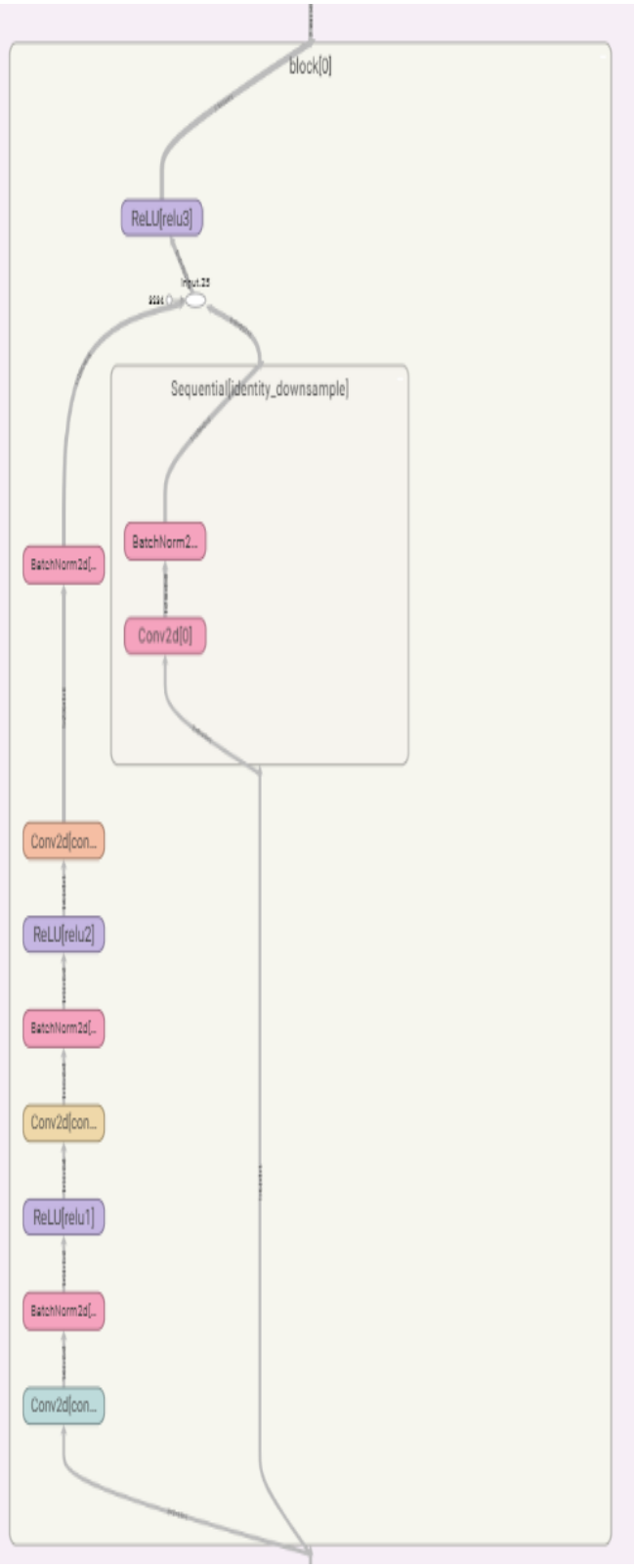

Fig. 44. ResNet Building Block

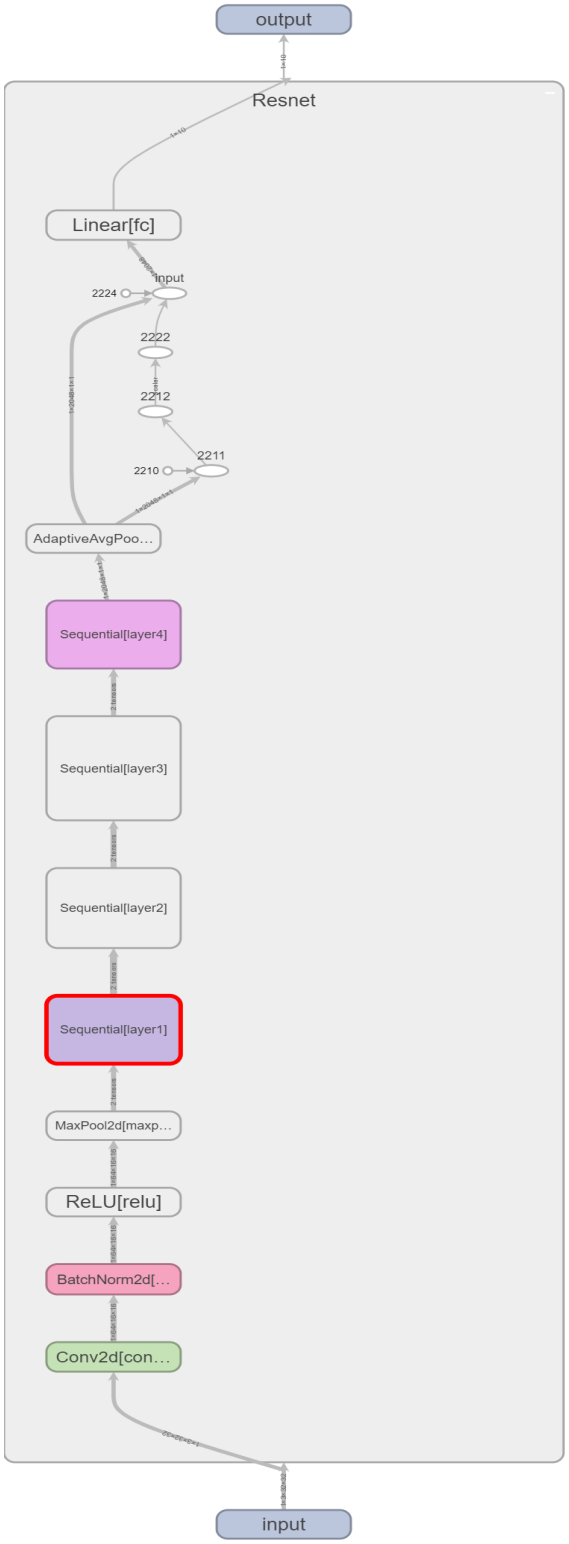

Fig. 45. ResNet Architecture

For 10 epochs and 20 mini-batch size, Training accuracy of 78.262 Percent was received. For testing, the accuracy was 66.06 Percent. The number of parameters are 23555082. Adam optimizer was used, with 1e-3 learning rate. Figures 45-47 presents the accuracy over train and test sets, along with loss over train set.

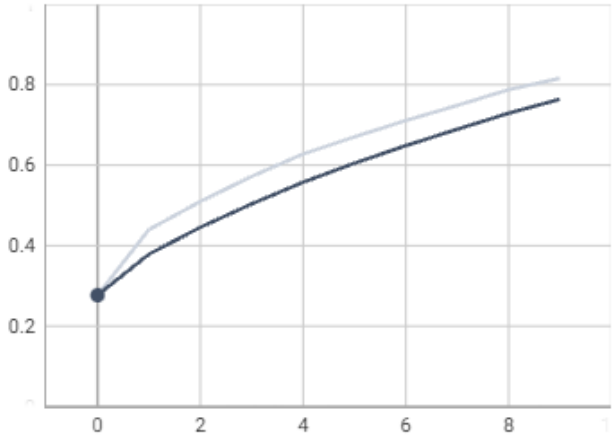

Fig. 46. Accuracy over Epochs on train set of ResNet Model.

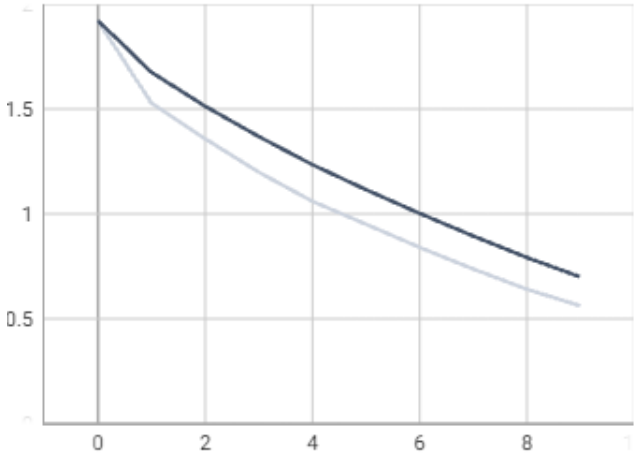

Fig. 47. Loss over Epochs on train set of ResNet Model.

1	0	1	2	3	4	5	6	7	8	9
0	4067	95	321	55	47	29	18	47	117	204
1	53	4730	24	15	2	2	18	15	14	127
2	160	28	3960	133	223	137	162	126	21	50
3	54	56	403	3233	124	550	241	190	23	121
4	100	21	553	130	3361	129	156	510	12	28
5	17	17	259	997	125	3072	96	366	6	45
6	14	91	262	221	120	45	4165	33	10	39
7	28	7	153	82	89	117	21	4449	2	52
8	214	299	72	64	19	13	10	20	4031	258
9	81	675	30	30	6	11	21	62	21	4063

Table 5. Confusion Matrix of ResNet Model on Training set (Accuracy - 78.262 Percent)

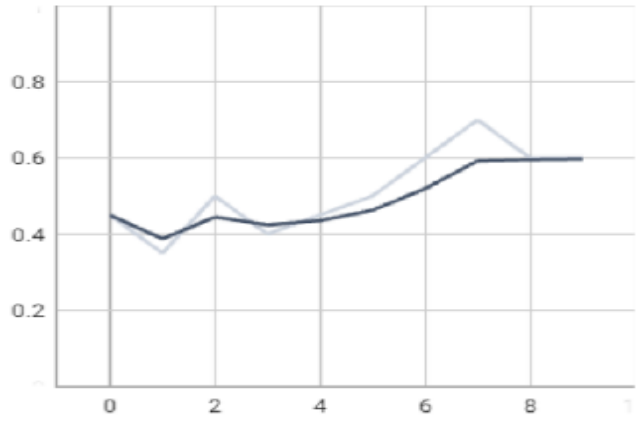

Fig. 48. Accuracy over Epochs on test set of ResNet Model.

I	0	1	2	3	4	5	6	7	8	9
0	704	39	79	22	19	10	12	17	46	52
1	25	863	10	8	2	2	7	6	4	73
2	62	11	622	64	50	62	57	53	7	12
3	15	22	120	466	58	148	69	64	9	29
4	23	2	147	59	525	36	56	142	7	3
5	10	8	85	203	42	503	34	98	5	12
6	6	14	60	76	43	24	734	22	7	14
7	20	5	43	33	37	44	8	791	1	18
8	74	73	20	23	11	7	5	17	700	70
9	38	182	11	17	3	5	7	27	12	698

Table 6. Confusion Matrix of ResNet Model on Test set (Accuracy - 66.06 Percent)

11. DenseNet

The DenseNet architecture implemented is given in Figure 50. There are mainly two types of blocks, the dense block ,which is made up of number of bottleneck layers. Each bottleneck layer (Figure 49) is made up of primarily two convolutional layers, a 1*1 and a 3*3. The output from the second convolution is added to the concatenated to the input to the layer. In this architecture (Figure 50), there are 6,12,24,16 bottleneck layers inside the dense blocks. Also, there is a transition block (Figure 51), which is mainly made of 1*1 convolution, and a average pool, for downsampling between subsequent dense blocks.

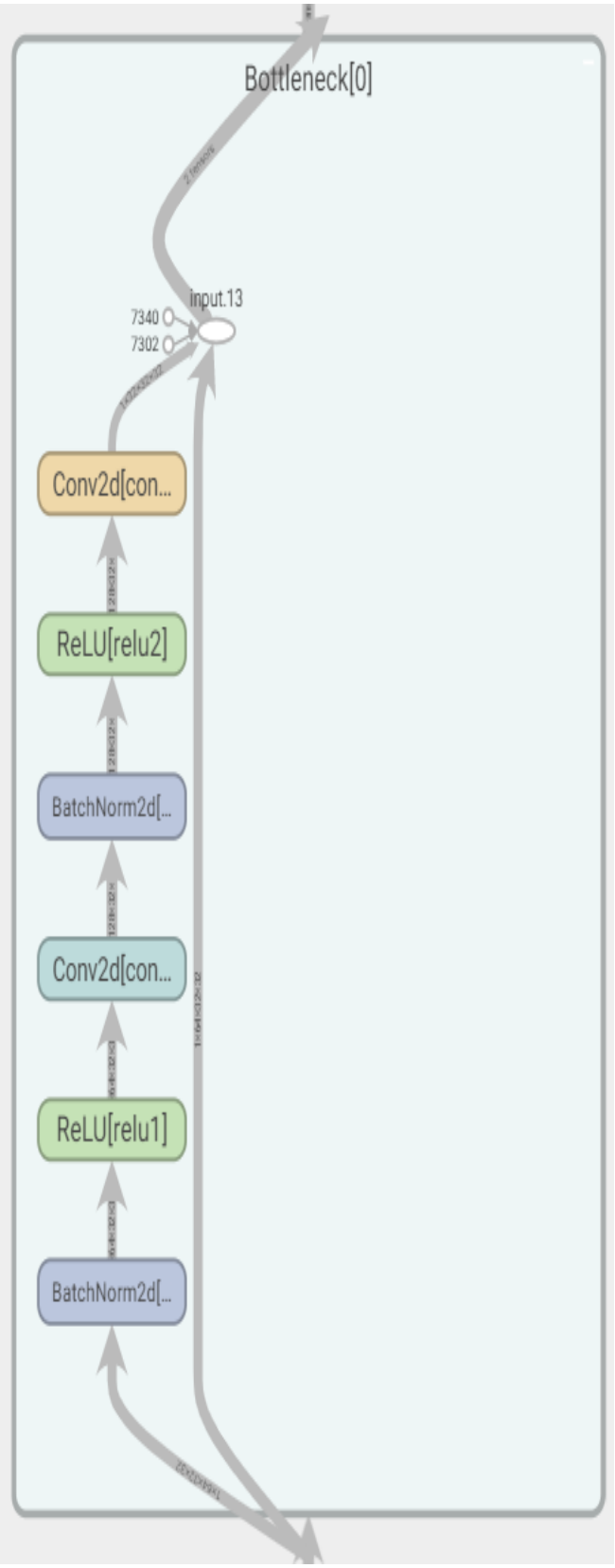

Fig. 49. DenseNet-Bottleneck Layer

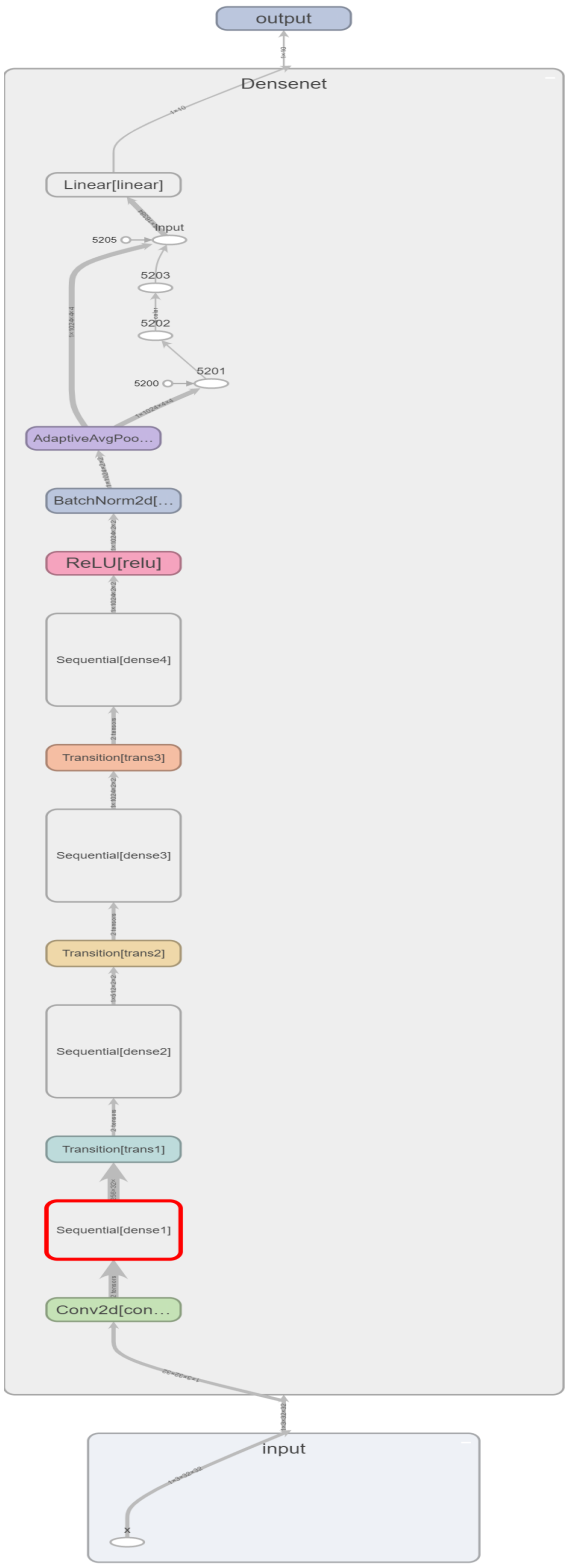

Fig. 50. DenseNet architecture

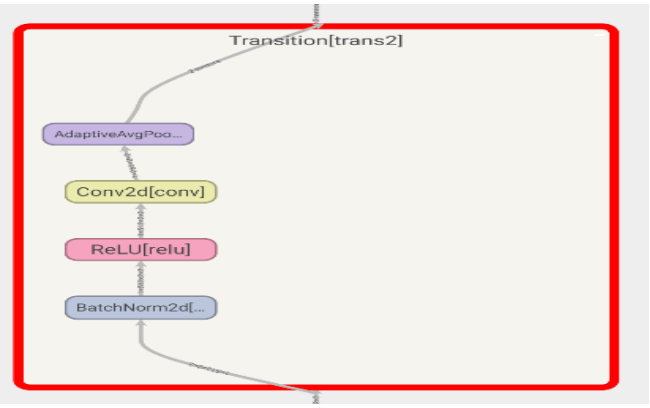

Fig. 51. DenseNet-Transition Layer

For 10 epochs and 20 mini-batch size, Train accuracy of 77.88 Percent and Test accuracy of 70.62 Percent was received. The number of parameters are 7109898. Adam optimizer was used, with 1e-3 learning rate. The confusion matrix and epoch, loss plots are included.

I	0	1	2	3	4	5	6	7	8	9
0	3880	36	377	56	105	21	12	77	371	65
1	49	4675	27	22	5	5	23	11	54	129
2	125	5	4042	99	389	67	113	101	53	6
3	46	13	674	2821	401	586	192	190	63	14
4	46	3	497	107	3857	57	66	330	33	4
5	14	1	593	831	250	2944	71	280	11	5
6	28	9	549	216	320	99	3725	33	17	4
7	31	3	266	92	230	140	6	4219	5	8
8	116	93	50	32	30	10	4	19	4591	55
9	108	400	47	40	24	11	9	103	72	4186

Table 7. Confusion Matrix of DenseNet Model on Training set (Accuracy - 77.88 Percent)

I	0	1	2	3	4	5	6	7	8	9
0	708	13	86	17	34	6	4	13	103	21
1	14	894	7	7	4	1	7	1	19	46
2	50	2	716	41	105	18	24	27	16	1
3	14	5	157	436	98	145	61	60	14	10
4	17	2	129	32	687	24	27	64	15	3
5	9	2	118	215	58	500	21	68	7	2
6	5	6	123	58	72	27	698	7	4	0
7	10	2	76	20	45	37	2	797	3	8
8	45	18	17	11	10	4	4	4	868	19
9	31	109	17	10	7	5	5	31	22	763

Table 8. Confusion Matrix of DenseNet Model on Test set (Accuracy - 70.62 Percent)

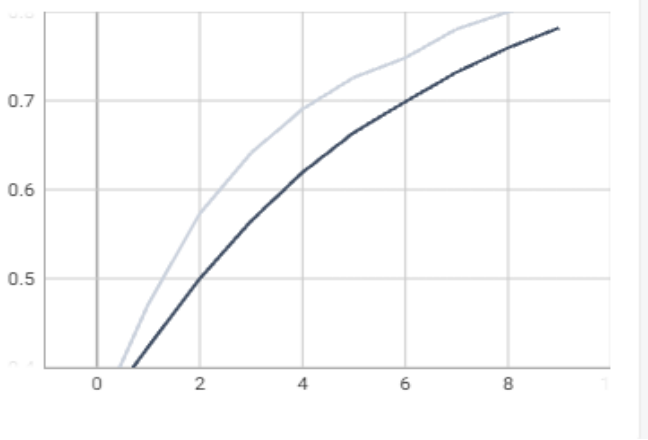

Fig. 52. Accuracy over Epochs on train set of DenseNet Model.

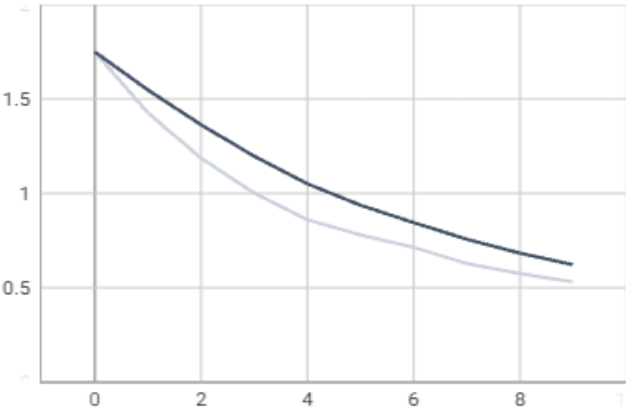

Fig. 53. Loss over Epochs on train set of DenseNet Model.

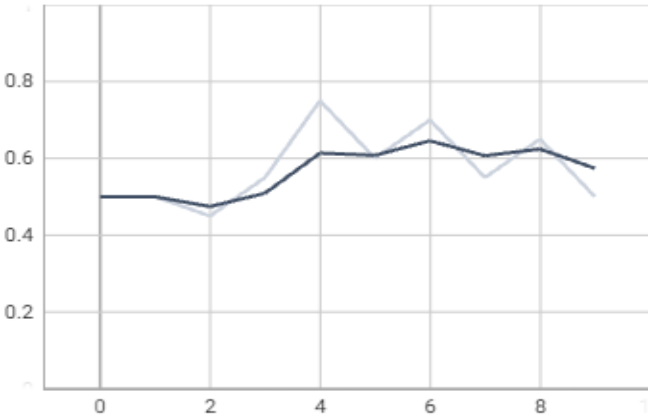

Fig. 54. Accuracy over Epochs on test set of DenseNet Model.

12. ResNeXt

The ResNeXt architecture implemented is given in Figure. 55. ResNeXt architecture is very much similar to ResNet architecture. Inside the ResNeXt block (Figure 56), the first three layers are the regular Convolution 1*1, BatchNorm and Relu. The second Convolution layer, 3*3, has an extra feature named cardinality, which groups the convolution layer with n number of similar layers. Then, going through layers of BatchNorm, Relu, Convolution 1*1 and BatchNorm, the output is addded to a downsampled input and then passed through Relu activation. In Figure 55, the four layer blocks contain 3, 4, 6 and 3 such ResNeXt blocks respectively.

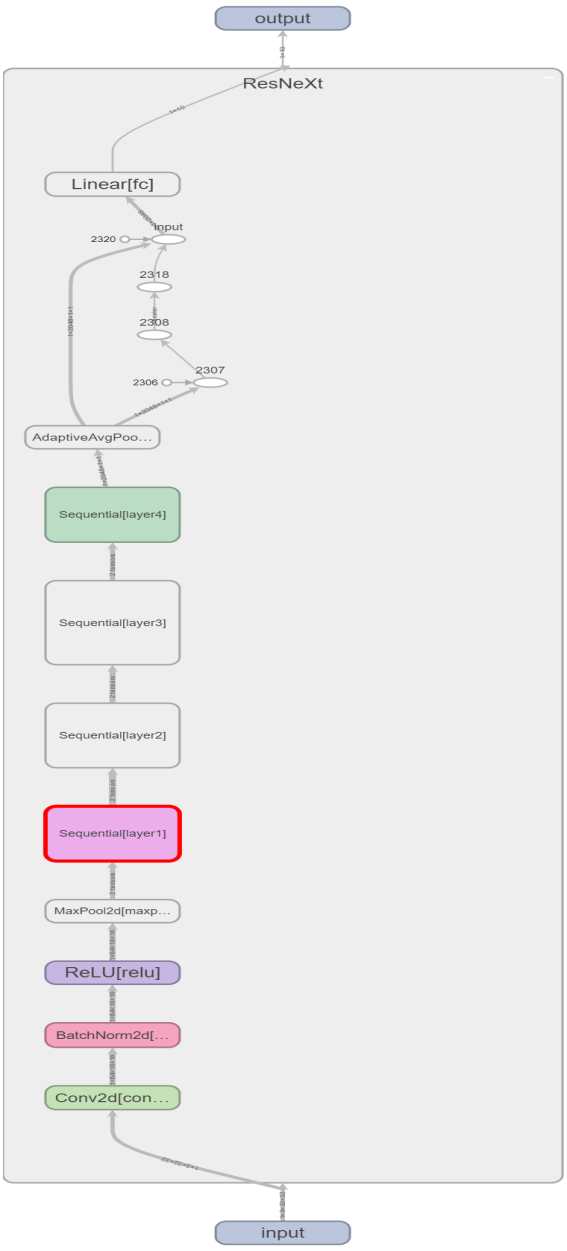

Fig. 55. ResNeXt Architecture

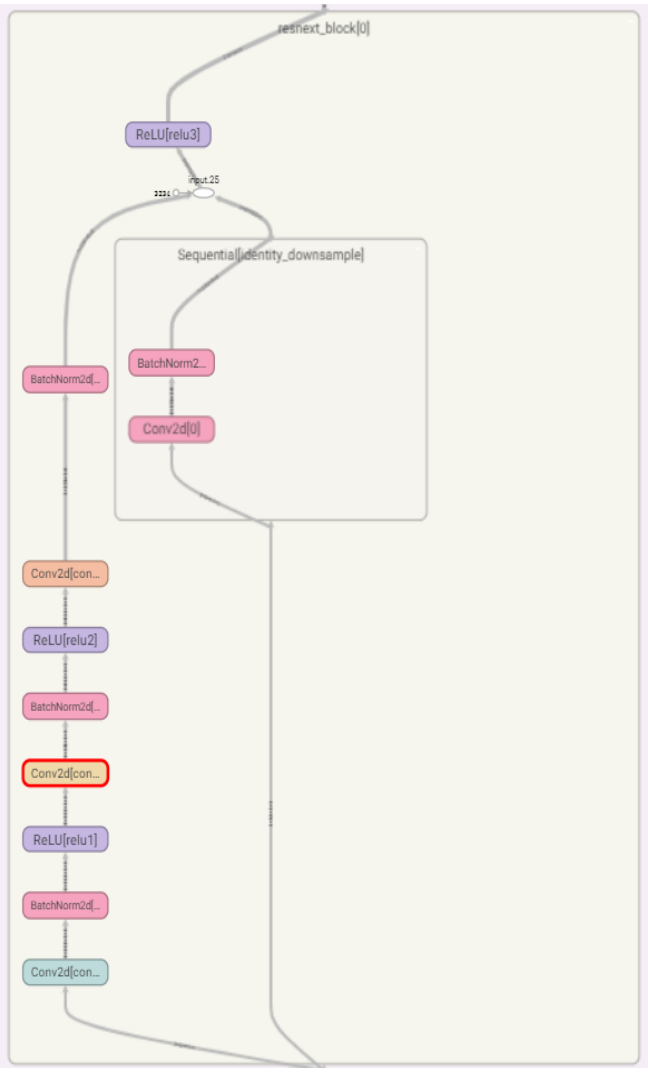

Fig. 56. ResNeXt Block

For 10 epochs and 20 mini-batch size, Train accuracy of 81.05 Percent and Test accuracy of 68.16 Percent was received. The number of parameters are 23034506. Adam optimizer was used, with 1e-3 learning rate. The confusion matrix and epoch, loss plots are included.

I	0	1	2	3	4	5	6	7	8	9
0	4193	23	166	95	67	13	17	32	206	188
1	58	4265	7	35	4	4	43	7	116	461
2	188	6	3790	304	167	114	274	79	43	35
3	51	9	180	3899	80	356	237	95	37	56
4	95	3	346	363	3553	112	236	229	37	26
5	12	3	183	1297	127	3061	109	163	16	29
6	8	9	115	314	75	37	4386	15	14	27
7	32	3	81	240	200	154	19	4223	11	37
8	149	38	26	98	18	14	12	3	4555	87
9	49	111	12	83	13	20	28	24	60	4600

Table 9. Confusion Matrix of ResNeXt Model on Train set (Accuracy - 81.05 Percent)

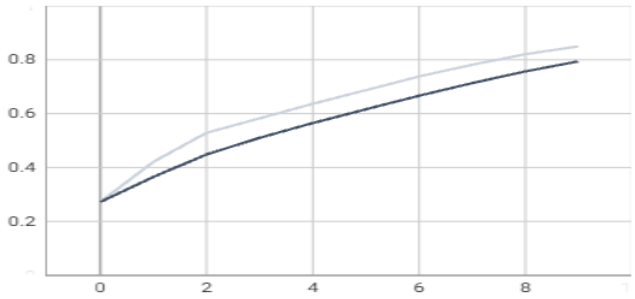

Fig. 57. Accuracy over Epochs on train set of ResNeXt Model.

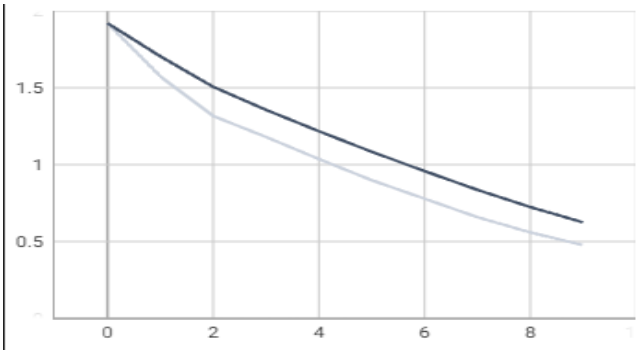

Fig. 58. Loss over Epochs on train set of ResNeXt Model.

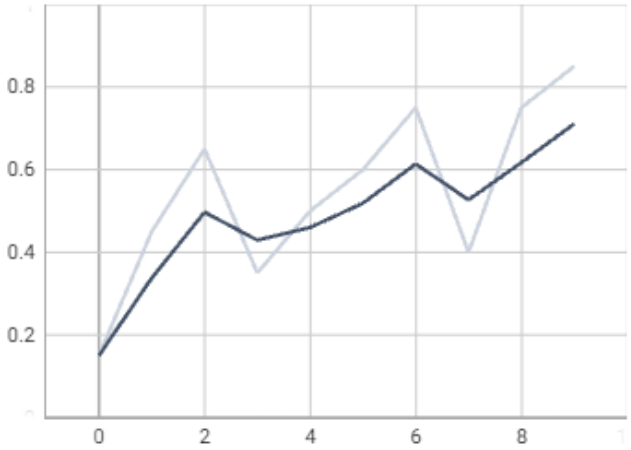

Fig. 59. Accuracy over Epochs on test set of ResNeXt Model.

I	0	1	2	3	4	5	6	7	8	9
0	700	12	49	38	30	6	8	10	86	61
1	25	725	3	13	3	4	12	2	41	172
2	61	4	605	98	52	48	79	26	10	17
3	22	5	62	620	46	98	74	37	14	22
4	31	1	121	107	537	33	70	82	10	8
5	9	0	57	290	30	491	39	61	7	16
6	8	2	32	103	25	13	784	12	9	12
7	17	4	32	62	52	68	8	734	4	19
8	74	20	9	38	10	4	4	3	803	35
9	33	56	10	25	3	5	9	13	29	817

Table 10. Confusion Matrix of ResNeXt Model on Test set (Accuracy - 68.16 Percent)

13. Analysis

The improved model from my first model definitely gave better results than the first one. However, more hyperparameter tuning could have been done to get more better results than the first model. DenseNet gave best results for me. However, it still is not that great. Increasing the number of epochs can be a possible solution for increasing train accuracy, and subsequently test accuracy for all the models, as they have been only trained on 10-20 epochs.

Network	Parameter Number	Train Accuracy	Test Accuracy	Inference runtime(micro s)
First	21482	59.622	55.66	0.039
Improved	21526	63.642	58.41	0.0367
ResNet	23555082	78.262	66.06	4.84
DenseNet	7109898	77.88	70.62	23.4
ResNeXt	23034506	81.05	68.16	5

Table 11. Comparison between the networks.