Assignment 3

S M Rakibur Rahman

Question 1

Part 1

Here, I have implemented a regular CNN where the number of filters in each layer increases as the depth of the network grows. The number of filters in the 10 convolution layers are 6, 12, 16, 24, 32, 48, 56, 64, 100, and 128.

The hyperparameters are learning rate, batch size and the choice of optimizer. I have tried the following:

learning rate : 0.01, 0.001, 0.0001

batch size: 32, 64, 128

the choice of optimizer: ‘SGD’, ‘adam’, ‘RMSProp’

The results are tabulated below:

SGD:

Learning rate	Batch size	Training loss	Training accuracy	Test loss	Test accuracy
.01	32	2.2884	0.1406	39.0145	0.3802
.01	64	2.3002	0.1129	2.2933	0.1077
.01	128	2.3012	0.1130	2.2981	0.1133
.001	32	2.3019	0.1123	2.3148	0.1166
.001	64	2.3019	0.1151	2.2956	0.1138
.001	128	2.3023	0.1147	2.3106	0.0935
.0001	32	2.3024	0.1141	2.3038	0.1164
.0001	64	2.3025	0.1369	2.3143	0.0937
.0001	128	2.3025	0.1354	2.3117	0.1014

Adam:

Learning rate	Batch size	Training loss	Training accuracy	Test loss	Test accuracy
.01	32	2.3029	0.1090	2.3015	0.1135
.01	64	2.3023	0.1095	2.3019	0.1135
.01	128	0.3513	0.8866	84.4337	0.7615
.001	32	0.2193	0.9277	14.6356	0.9684
.001	64	0.2201	0.9281	10.9096	0.9773
.001	128	0.2915	0.9075	11.0025	0.9743
.0001	32	0.4084	0.8687	20.4804	0.9525
.0001	64	0.5420	0.8247	29.5571	0.9356
.0001	128	0.8547	0.7179	39.2760	0.9120

RMSProp:

Learning rate	Batch size	Training loss	Training accuracy	Test loss	Test accuracy
.01	32	2.4258	0.1093	2.5493	0.1032
.01	64	2.9218	0.1099	2.3213	0.1139
.01	128	2.3298	0.1095	2.3019	0.1135
.001	32	0.2236	0.9273	12.9595	0.9756
.001	64	0.2503	0.9184	18.9364	0.9676
.001	128	0.3080	0.9002	17.3935	0.9646
.0001	32	0.3927	0.8793	22.5251	0.9533
.0001	64	0.6077	0.8048	25.5463	0.9385
.0001	128	0.8413	0.7347	48.9262	0.8852

Observations: We observe that performance of Adam and RMSProp optimizer was comparable. Adam performed slightly better. SGD optimizer performed worst. Learning rate .01 produced significantly worst performance compared to others and learning rate .001 produced the best performance. Batch size 128 performed worst and performance of batch size 32 and 64 was comparable. Batch size 32 performed slightly better overall.

Therefore, we conclude that for the CNN network in part 1, Adam optimizer with batch size of 32 and learning rate 0.001 is the best choice.

Part 2

Here, I have implemented an inverted CNN where the number of filters in each layer decreases as the depth of the network grows. The number of filters in the 10 convolution layers are 128, 100, 64, 56, 48, 32, 24 ,16, 12, and 6.

The hyperparameters are learning rate, batch size and the choice of optimizer. I have tried the following:

learning rate : 0.01, 0.001, 0.0001

batch size: 32, 64, 128

the choice of optimizer: ‘SGD’, ‘adam’, ‘R3MSProp’

The results are tabulated below:

SGD:

Learning rate	Batch size	Training loss	Training accuracy	Test loss	Test accuracy
.01	32	2.3011	0.1120	2.2996	0.1135
.01	64	2.3007	0.1123	2.3092	0.0958
.01	128	2.3018	0.1123	2.3017	0.1121
.001	32	2.3020	0.1146	2.3036	0.1391
.001	64	2.3022	0.1134	2.2997	0.0942
.001	128	2.3023	0.1126	2.2992	0.1198

.0001	32	2.3025	0.1110	2.3045	0.0666
.0001	64	2.3026	0.1142	2.3053	0.1028
.0001	128	2.3025	0.1089	2.3083	0.0224

Adam:

Learning rate	Batch size	Training loss	Training accuracy	Test loss	Test accuracy
.01	32	2.3042	0.1090	4.4671	0.1218
.01	64	0.6686	0.7624	31.2305	0.1123
.01	128	2.3090	0.1117	2.3275	0.1040
.001	32	0.3177	0.9001	20.4333	0.9421
.001	64	0.2988	0.9032	12.8614	0.9715
.001	128	0.3814	0.8759	20.0418	0.9600
.0001	32	0.5216	0.8348	31.2361	0.9346
.0001	64	0.6986	0.7790	42.6318	0.9023
.0001	128	0.9086	0.6927	42.6722	0.9097

RMSProp:

Learning rate	Batch size	Training loss	Training accuracy	Test loss	Test accuracy
.01	32	2.3976	0.1089	2.3123	0.1133
.01	64	2.8354	0.1104	2.3032	0.1028
.01	128	2.3131	0.1097	2.3013	0.1135
.001	32	0.2705	0.9133	13.1042	0.9762
.001	64	0.3131	0.9018	17.3527	0.9710
.001	128	0.6246	0.7848	26.9330	0.9370
.0001	32	0.8068	0.7227	48.2582	0.8990
.0001	64	0.8385	0.7244	50.3453	0.8960
.0001	128	0.9003	0.7078	49.3432	0.8971

Observations: We observe that performance of Adam and RMSProp optimizer was comparable. SGD optimizer performed worst. Learning rate .01 produced significantly worst performance compared to others and learning rate .001 produced the best performance. Batch size 128 performed worst and performance of batch size 32 and 64 was comparable. Batch size 32 performed slightly better overall. Overall, there is very little to differentiate between Adam and RMSProp. RMSProp with learning rate .001 and batch size 32 performed best and appears to be best choice/

Therefore, we conclude that for the CNN network in part 2, RMSProp optimizer with batch size of 32 and learning rate 0.001 is the best choice.

Part 3

Here, I have implemented An hour-glass shaped CNN where the number of filters will increase till the Lth layer and reduce afterwards.. The number of filters in the 10 convolution layers are 6, 12, 16, 24, 32, 48, 56, 64, 100, and 128.

The hyperparameters are learning rate, batch size and the choice of optimizer. I have tried the following:

learning rate : 0.01, 0.001, 0.0001

batch size: 32, 64, 128

the choice of optimizer: ‘SGD’, ‘adam’, ‘RMSProp’

The results are tabulated below:

SGD:

Learning rate	Batch size	Training loss	Training accuracy	Test loss	Test accuracy
.01	32	2.3001	0.1124	2.2754	0.1381
.01	64	2.2996	0.1124	2.2825	0.2389
.01	128	2.3014	0.1127	2.3033	0.1506
.001	32	2.3019	0.1120	2.3053	0.1099
.001	64	2.3023	0.1194	2.3024	0.1081
.001	128	2.3023	0.1288	2.3051	0.1614
.0001	32	2.3025	0.1145	2.3057	0.1254
.0001	64	2.3025	0.1321	2.3041	0.0882
.0001	128	2.3025	0.1374	2.3177	0.1101

Adam:

Learning rate	Batch size	Training loss	Training accuracy	Test loss	Test accuracy
.01	32	2.3039	0.1088	2.3020	0.1135
.01	64	2.3042	0.1090	2.3013	0.1135
.01	128	0.4361	0.8519	20.0621	0.9448
.001	32	0.2420	0.9235	10.5014	0.9733
.001	64	0.2803	0.9069	11.7000	0.9734
.001	128	0.4027	0.8702	15.1281	0.9675
.0001	32	0.5667	0.8188	32.0731	0.9341
.0001	64	0.7232	0.7540	37.8643	0.9211
.0001	128	0.9990	0.6615	51.3646	0.8836

RMSProp:

Learning rate	Batch size	Training loss	Training accuracy	Test loss	Test accuracy
.01	32	2.3107	0.1093	2.3024	0.1135
.01	64	2.3025	0.1095	2.3018	0.1135
.01	128	2.3022	0.1110	2.3016	0.1135
.001	32	0.2543	0.9202	15.1433	0.9716
.001	64	0.2634	0.9154	11.1377	0.9737
.001	128	0.3691	0.8763	13.7451	0.9676
.0001	32	0.6538	0.7828	38.4760	0.9077
.0001	64	1.0127	0.6713	60.7841	0.8668
.0001	128	1.1133	0.6250	58.7395	0.8531

Observations: We observe that performance of Adam and RMSProp optimizer was comparable. SGD optimizer performed worst. Learning rate .01 produced significantly worst performance compared to others and learning rate .001 produced the best performance. Batch size 128 performed worst and performance of batch size 32 and 64 was comparable. Batch size 32 performed slightly better overall. Overall, there is very little to differentiate between Adam and RMSProp when learning rate is .001. However, Adam performed better when learning rate was .0001 and for both the optimizers batch size 64 performed better.

Therefore, we conclude that for the CNN network in part 3, Adam optimizer with batch size of 64 and learning rate 0.001 is the best choice.

Question 2

The LeNet CNN network is implemented using Keras. Batch size 32 is kept fixed and learning rate is varied to observe its effects

1. What is the effect of learning rate on the training process? Which performed best?

Learning rate	Training loss	Training accuracy	Test loss	Test Accuracy
0.1	2.3152	0.1002	2.3141	0.100
0.01	2.3031	0.0999	2.3031	0.099
0.001	0.0947	0.9682	739.4534	0.5449
0.0001	0.6549	0.7746	261.933	0.5034
0.00001	1.3239	0.5357	250.678	0.3968

We observe that performance improved when learning rate was increased from 0.00001 to 0.001. But after that performance degraded with increase in learning rate. At .00001 and 0.0001 learning rates the network wasn’t learning much. 0.0001 and 0.00001 learning rates were too high and network couldn’t reach the optimum. 0.001 was the optimum learning rate which yielded the best performance.

2. What is the effect of batch size on the training process? Which performed best?

Here, learning rate 0.001 is kept fixed and batch size varied

Batch size	Training loss	Training accuracy	Test loss	Test Accuracy
32	0.1055	0.9654	850.9072	0.5570
64	0.0836	0.9707	842.4243	0.5365
128	0.0669	0.9767	910.7155	0.4988
256	0.1571	0.9482	518.7839	0.5333
500	0.4877	0.8333	276.3285	0.5433

Test accuracy for all the batch sizes is similar. There is no clear trend. Considering both training and test accuracy, batch size 32 performed best.

3. Try different hyperparameters to obtain the best accuracy on the test set. What is your

best performance and what were the hyperparameters?

The hyperparameters are optimizer, batch size and learning rate. I tried RMS Prop with batch size 32, learning rate 0.001 and obtained the following results:

Training loss: 0.0930
Training accuracy: 0.9725
Test loss: 925.0296020507812
Test accuracy: 0.5303000211715698

Its performance is comparable to Adam but slightly worse. I tried .005 and .0005 which are around the optimum learning rate found in part 1 and the results are following

Learning rate	Training loss	Training accuracy	Test loss	Test Accuracy
0.005	0.0794	0.9730	698.4704	0.5840
0.0005	2.3034	0.0992	2.397	0.099

Performance of .005 is better than 0.001. So, the best performance obtained is test accuracy 0.5840 with batch size 32, learning rate 0.005 and Adam optimizer

4. Implement an equivalent feed forward network for the same task with each hidden layer containing the same number of neurons as the number of filters in each convolution layer. Use the 'Adam' optimizer to train your network on the CIFAR-10 dataset for a fixed set of 25 epochs. Compare its performance with your LeNet implementation based on the following questions:

An equivalent feed forward neural network for the same task with each hidden layer containing the same number of neurons as the number of filters in each convolution layer is implemented in keras. The hidden layers of the NN are added in the model as dense layers in Keras. 'Adam' optimizer is used to train the NN network on the CIFAR-10 dataset for a fixed set of 25 epochs

a. What is its performance?

Its performance is significantly worse than the LeNet CNN network. Performance of the NN

Training loss: 2.3033 Training accuracy; 0.0994
Test loss: 2.3032. Test accuracy: 0.10000

b. How many parameters are there in this network compared to the LeNet implementation? Are they worth it?

Number of total parameters are

The feed forward NN	21,800
LeNet Network	697,046

So, the feed forward NN has around 32 times less parameters than the LeNet network.

They (the feed forward NN) are not worth it. Because, they have significantly poorer performance compared to the LeNet.

Answer to question 3 is in next page

behavior is neglected. We

Ans To the Q No-3

Given $X = \begin{bmatrix} 7 & 5 & 0 & 0 & 3\cancel{0} & 2 \\ 6 & 4 & 5 & 1 & 4 & 8 \\ 9 & 0 & 2 & 2 & 5 & 4 \\ 6 & 3 & 4 & 7 & 9 & 8 \\ 5 & 7 & 5 & 6 & 9 & 0 \\ 7 & 9 & 0 & 8 & 2 & 3 \end{bmatrix}$ $f = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$

1. Given depth of input is 1

Input has dimensions $6 \times 6 \times 1$

dimension of filter 3×3

Here filter dimension $F = 3$

Depth $C = 1$

Number of filter $K = 1$

Number of parameter in the kernel $= F^2CK + K$

$$= 3^2 \times 1 \times 1 + 1$$

$$= 9 + 1 = 10$$

2. We know, output size without padding $= \frac{N-F}{Stride} + 1$

Here $N =$ input size $= 6$

$F =$ filter size $= 3$

Choosing stride $= 1$

$\therefore$ So, output size $= \frac{6-3}{1} + 1 = 3 + 1 = 4$

metrical nonlinearity of large

So the output size is 4X4X1

Convolution operation is performed by sliding the kernel over the input. First, the kernel is placed on the top left 3X3 block of the input. Elementwise multiplication is done and then it is summed up.

7X1	5X0	0X-1	0	3	2
6X2	4X0	5X-2	1	4	8
9X1	0X0	2X-1	2	5	4
6	3	4	7	9	8
5	7	5	6	9	0
7	9	0	8	2	3

$$7\times1+5\times0+0(-1)+6\times2+4\times0+5(-2)+9\times1+0\times0+2(-1)$$
$$=16$$

Then the filter will slide as shown below and similar operation is performed

7	5X1	0X0	0X-1	3	2
6	4X2	5X0	1X-2	4	8
9	0X1	2X0	2X-1	5	4
6	3	4	7	9	8
5	7	5	6	9	0
7	9	0	8	2	3

$$5\times1+0\times0+0(-1)+4\times2+5\times0+1(-2)+0\times1+2\times0+2(-1)$$
$$=9$$

After completing all the convolution operation we get the output activation map

16	9	-4	-18
17	-5	-10	-12
11	-9	-17	2
9	-1	-15	16

3

We apply a 2x2 max pooling. This operation involves getting maximum of 2x2 blocks and reduce the dimension. So, first we consider top left 2x2 block

16	9
17	-5

maximum is 17

Then the next

-4	-18
-10	-12

maximum is -4 and so on

After maxpooling, output is

17	-4
11	16