

HW4:



Submitting: Jeremie Sasson 336319470 Rakefet Rozen 034148874

Part1:

First model summary:

Layer (type)	Output Shape	Param #
=====		
flatten (Flatten)	(None, 1024)	0
=====		
dense (Dense)	(None, 300)	307500
=====		
dense_1 (Dense)	(None, 150)	45150
=====		
dense_2 (Dense)	(None, 4)	604
=====		

Total params: 353,254

Trainable params: 353,254

Non-trainable params: 0

1. Loss and accuracy of first model with RELU activation function batch_size 64 epochs 25:
loss: 0.8075, acc: 0.6629
2. Loss and accuracy of second model with sigmoid activation function batch_size 64 epochs 25:
loss: 1.0690, acc: 0.56




The sigmoid activation function gave a 0.56 accuracy while the RELU 0.66. The sigmoid activation function has a disadvantage that its derivative is always smaller than one and gets small for large absolute values. Moreover, when you have several layers the result is a multiplication of all results meaning the number goes very fast to zero and the Layers learning effect cannot be used. In the RELU function only negative results goes to zero and all others 1. Meaning you can use as many layers as you want and still have some gradient multiplications different from zero and equal to 1.

3. Loss and accuracy of second model with sigmoid activation function batch_size 64 epochs 25:
loss: 1.0690, acc: 0.56

Loss and accuracy of second model with sigmoid activation function batch_size 64 epochs 40:
loss: 0.9011, acc: 0.6343


Each epoch trains all training data on the neural network. Each run improves the weights. Running the data only once will build a network that underfits our data set. On the other hand, running our data set too many times will create an over fit. But finding the midway will improve the network's performance. In this case 40 epochs resulted a better accuracy and loss then 25 epochs on the test set, meaning we are still not overfitted and the network has improved.

4. Loss and accuracy of first model with RELU activation function batch_size 32 epochs 50:
loss: 0.8707, acc: 0.6457.

 The advantages of mini-batches vs. SGD is that it accurately updates the parameters.

5. Loss and accuracy of first model with RELU activation function batch_size 32 epochs 50 with batch normalization:

loss: 0.8635, acc: 0.6800.

 Batch normalization improved the accuracy from 0.6457 to 0.68. The normalization process brings our data back to the same distribution and by that helps the training of deep networks. It also accelerates the learning process by using higher learning rate.


PART2:

Have a look at the model below and answer the following:


6. How many layers does it have?

Answer: 8 layers

7. How many filter in each layer?

Answer: 64, 128, 128, 256, 256, 512, 128, 4 

8. Would the number of parameters be similar to a fully connected NN?

If the question referred to the same number of layers with the same number of neurons but fully connected than the answer is that in a fully connected network we would have created many more parameters as for each data point would be connected in each layers to all neurons at that layer, while in a convolution layer the data point are spread in between the neurons meaning less parameters are needed. 

If the question referred to a comparison between this assignment first fully connected network in part1 and the convolution network given at part2, than the answer is that the convolution network includes more parameters. It is a bigger net and includes more layers as can be seen from the summary of the nets: part1 network includes about 300000 parameters while the part 2 includes about 3 million parameters.

Part2 network summary:

Layer (type)	Output Shape	Param #
=====		
permute_1 (Permute)	(None, 32, 32, 1)	0
<hr/>		
Conv2D_1 (Conv2D)	(None, 32, 32, 64)	640
<hr/>		
dropout_6 (Dropout)	(None, 32, 32, 64)	0
<hr/>		
batch_normalization_v1_7 (Ba	(None, 32, 32, 64)	128

max_pooling2d_3 (MaxPooling2)	(None, 16, 16, 64)	0
-------------------------------	--------------------	---

Conv2D_2 (Conv2D)	(None, 16, 16, 128)	73856
-------------------	---------------------	-------

dropout_7 (Dropout)	(None, 16, 16, 128)	0
---------------------	---------------------	---

batch_normalization_v1_8 (Ba	(None, 16, 16, 128)	64
------------------------------	---------------------	----

Conv2D_3 (Conv2D)	(None, 16, 16, 128)	147584
-------------------	---------------------	--------

dropout_8 (Dropout)	(None, 16, 16, 128)	0
---------------------	---------------------	---

batch_normalization_v1_9 (Ba	(None, 16, 16, 128)	64
------------------------------	---------------------	----

max_pooling2d_4 (MaxPooling2)	(None, 8, 8, 128)	0
-------------------------------	-------------------	---

Conv2D_4 (Conv2D)	(None, 8, 8, 256)	295168
-------------------	-------------------	--------

dropout_9 (Dropout)	(None, 8, 8, 256)	0
---------------------	-------------------	---

batch_normalization_v1_10 (B	(None, 8, 8, 256)	32
------------------------------	-------------------	----

Conv2D_5 (Conv2D)	(None, 8, 8, 256)	590080
-------------------	-------------------	--------

dropout_10 (Dropout)	(None, 8, 8, 256)	0
----------------------	-------------------	---

batch_normalization_v1_11 (B	(None, 8, 8, 256)	32
------------------------------	-------------------	----

max_pooling2d_5 (MaxPooling2)	(None, 4, 4, 256)	0
-------------------------------	-------------------	---

flatten_2 (Flatten)	(None, 4096)	0
<hr/>		
FCN_1 (Dense)	(None, 512)	2097664
<hr/>		
dropout_11 (Dropout)	(None, 512)	0
<hr/>		
FCN_2 (Dense)	(None, 128)	65664
<hr/>		
FCN_3 (Dense)	(None, 4)	516
<hr/>		
=====		

Total params: 3,271,492

Trainable params: 3,271,332

Non-trainable params: 160

9. Is this specific NN performing regularization?

Yes it does with a l2 function and also the dropout is considered a regularization.

10. Part2 network with filters: [64, 128, 128, 256, 256]: **Loss: 6.73, acc: 0.382**

Part2 network with filters: [32, 64, 64, 128, 128]: **Loss: 4.18, acc: 0.382**

I would expect the accuracy of the bigger net to be better, but the accuracy results are the same in both set of filters. The loss function is better with the half-sized filters. These results may indicate that we over fitted our train set with the big number of neurons and that a smaller net is actually better generalizes the data.