HW 4 – Theory Questions

Part 1

Task 1:

```
test loss, test acc: [0.8250346779823303, 0.6742857098579407]
```

Task 2[1]:

The model can affect the efficiency of computer computation as well as the inference results. For example: RELU is very efficient and fast in calculations, but when the input values are approximately zero or negative, it is impossible to calculate backpropagation, which means there is no training. At the same time, the leaky RELU allows training through backpropagation, but the prediction with negative input data is imprecise.

Other more accurate models of activation functions are sigmoid and hyperbolic tangent. They give smooth gradients and accurate predictions (the hyperbolic tangent is also centered around zero, making it easier to compute for inputs that are very negative, very positive, or neutral in value). However, these features also have their drawbacks. Such models are computationally heavy, and at their boundaries (very large or very small values on the x-axis) lead to gradient vanishing, because at these ends the gradient change is almost zero. This can lead to slow predictions.

In our case, we expect to get slightly more accurate results for the sigmoid model, but we get worse results. Perhaps, our initial weights were unsuccessful (they were on the sides of the possible values, where the gradient is very small).

Task 3[2]:

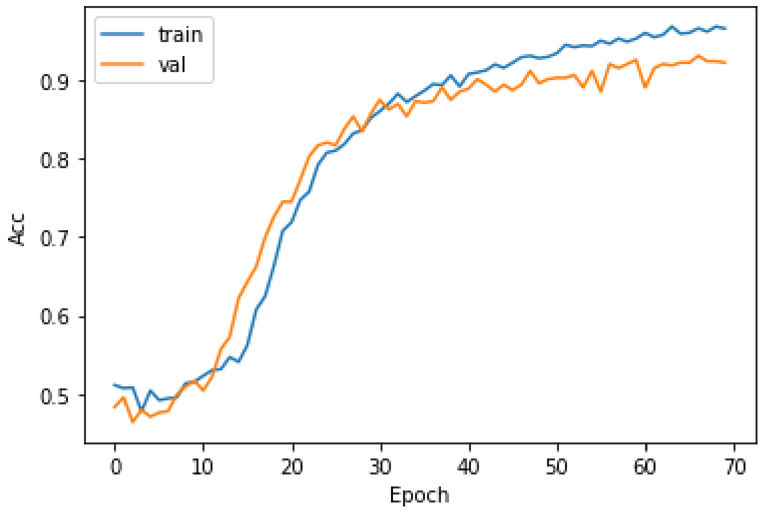
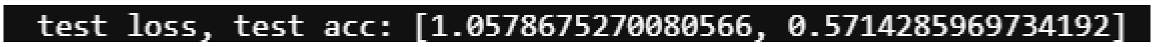


Figure 1

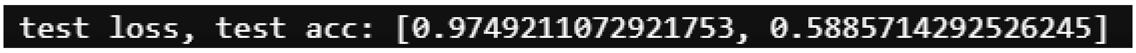
The number of epochs affects the performance. More epochs mean that we will go through the train set more times, during learning. When there is a small number of epochs, the influence is significant and you can see a strong improvement in the performance (see Figure 1), but at some point, the system comes to a state of equilibrium and an increase in the number of epochs has almost no effect on the results and can lead to overfitting. In our case, as we can see, there was no significant improvement.

25 epochs:



```
test loss, test acc: [1.0578675270080566, 0.5714285969734192]
```

40 epochs:



```
test loss, test acc: [0.9749211072921753, 0.5885714292526245]
```

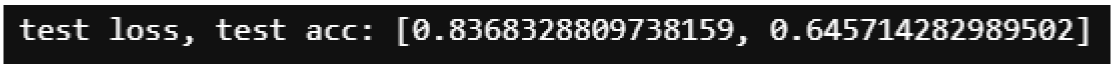
Task 4[3] Mini-Batches:

To begin with, let's define the disadvantages of the SGD. Steps to a minimum are very noisy and not directional. And because of the noise, it can take longer to reach the minimum of loss function. Stochastic gradient descent is harder to compute.

At the same time, a mini-batch, it is a mixture of SGD and a Batch gradient descent. It is more directional (its vectorization is better) so it is computationally simpler, and computations require less memory and less noisy. If this algorithm gets stuck in a local minimum, then some noisy steps can cause it to break out of the local minimum.

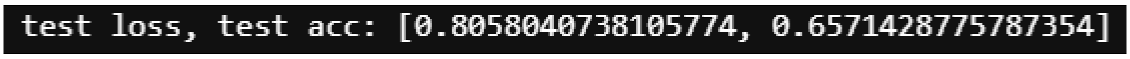
Pay attention, that reducing the batch size means that the gradient steps are smaller for the same number of samples. In our case, when the number of batches is bigger we get little better results.

32:



```
test loss, test acc: [0.8368328809738159, 0.645714282989502]
```

64:



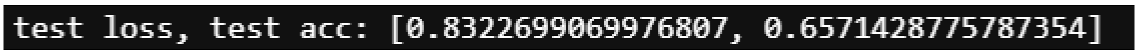
```
test loss, test acc: [0.8058040738105774, 0.6571428775787354]
```

Task 4 – Batch normalization [4]:

Batch normalization is normalizing the mean and variance of each feature at every level of representation during training. It enables to accelerate the learning process due to using higher learning rates and helps with training of very deep networks.

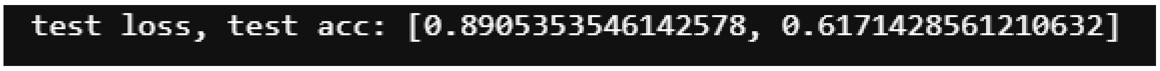
We expected to get better results, but the results we get are worser sometimes than without normalization. It could be because our data is not very good.

No BATCH_NORMALIZATION:



```
test loss, test acc: [0.8322699069976807, 0.6571428775787354]
```

With BATCH_NORMALIZATION:



```
test loss, test acc: [0.8905353546142578, 0.6171428561210632]
```

Part 2

Task 1: 2D CNN

How many layers does it have? It has 8 layers (5 convolutional layers + 3 Dense).

How many filters in each layer? [64, 128, 128, 256, 256]

Would the number of parameters be similar to a fully connected NN? The number of parameters will be bigger in fully connected NN because in fully connected NN every neuron in one layer connected to every neuron in another layer, while in CNN groups of neurons (depends on filter) have the same weight so we need to learn less. [5]

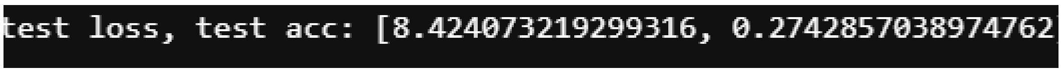
#params of CNN = ((shape of width of the filter * shape of height of the filter * number of filters in the previous layer+bias)*number of filters)[8]

#params of FC = ((current layer neurons c * previous layer neurons p)+bias*c)[8], where c is a current layer

Is this specific NN performing regularization? There is dropout and L2 regularization in the code. The dropout randomly drops out nodes during training and so provide regularization to reduce overfitting [6]. L2 – tries to reduce overfitting. L2 regularization is also known as weight reduction because it causes the weights to decrease to zero (but not exactly zero).[7]

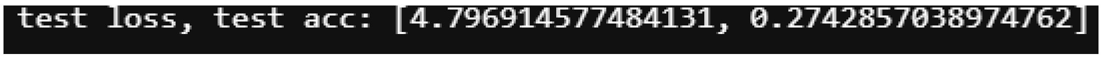
Task 2:

Initial number of filters:



```
test loss, test acc: [8.424073219299316, 0.2742857038974762
```

Reduced number of filters:



```
test loss, test acc: [4.796914577484131, 0.2742857038974762]
```

Reducing the number of filters leads to reducing complexity of the model. We can see that loss reduced, but the accuracy is not really affected from the reducing of filters. This phenomena is not constant we saw some changes, influenced by initial weights.

PAY ATTENTION:::
RESULTS IN IPYNB COULD DIFER FROM RESULTS IN THIS DOCUMENT, BECAUSE WE HAVE RAN THE CODE MANY TIMES

References:

[1] https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/

[2] https://www.researchgate.net/figure/Accuracy-versus-different-epoch-plot_fig3_327261283

[3] https://medium.com/@divakar_239/stochastic-vs-batch-gradient-descent-8820568eada1

[4] Lecture 16, slide 35.

[5] https://medium.com/swlh/fully-connected-vs-convolutional-neural-networks-813ca7bc6ee5

[6] https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

[7] https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

[8] https://towardsdatascience.com/understanding-and-calculating-the-number-of-parameters-in-convolution-neural-networks-cnns-fc88790d530d