

## HW4-MLH


Shachar Zigron 316356401 Shay Ohana 315800375

### PART 1: Fully connected layers

**Task 2:** "Change the activation functions to LeakyRelu or tanh or sigmoid. Name the new model new\_a\_model. Explain how it can affect the model."

**Answer:** In Relu activation function for negative input values the output value is 0 and for positive inputs the calculation is simple, by the identity function, i.e., the output is the same as the input. Whereas the sigmoid function requires a more complicated calculation.

In addition, due to the mode of operation of the Relu function some of the neurons are not activated, in contrast to the sigmoid where all the neurons are activated. Therefore, a change of the activation function from Relu to sigmoid will cause to higher computational cost and the computation time is longer.

In the sigmoid function for small or large input values (outside the  of -2 to 2) the output value will not change and therefore causes the gradient vanishing problem, and the model may stop learning or learn very slowly. Whereas in the Relu function the gradient will be zero only for the negative input values i.e. there is a larger range of input values where the gradient is significant and learning is performed and therefore we will get a different prediction for the different functions.

**Task 3:** "train the new model using 25 and 40 epochs. What difference does it makes in term of performance? "

**Answer:** In general, using a few epochs may cause to underfitting and apply the model on new data will yield poor performance i.e. we will get low accuracy and high loss function which means that the classification performed is not good enough. Whereas where using high number of epochs may result to overfitting to the training set and therefore when entering new data, we will get an incorrect classification due to limited generalization capability.

Epochs affects how much the model is able to adapt to new data("Generalization") and therefore by tuning the number of epochs as a hyperparameter we can find the number that leads to the best performances.

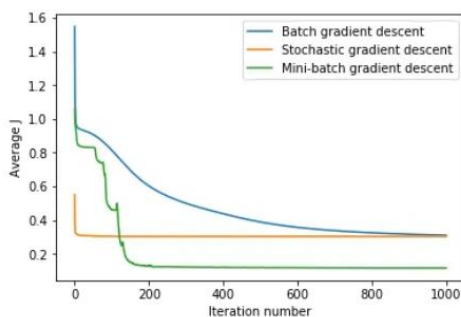
In our case the number of epochs is not high and therefore using a higher number of epochs will yield better performances (40 epochs), i.e. higher accuracy and lower loss function.

#### **Task 4: "What are the advantages of the mini-batch vs. SGD?"**

The SGD update algorithm performs a Gradient descent calculation for each of the given samples, i.e. for 1000 input values 1000 calculations will be performed and weights will be updated. This form of updating is sensitive to noise as each sample is considered.

In contrast the mini-batch update algorithm performs a Gradient descent calculation for a small number of samples together and then performs the update of the weights according to the mean gradient descent. The advantages of this algorithm over SGD are a smaller computation cost and therefore a shorter calculation time due to smaller number of update steps, is mainly reflected for a large data set. In addition, it is less sensitive to noise, since the calculation are over a set of data and not on a single sample.

In mini batch step size is larger so it is less likely to get stuck in a local minimum, therefore converges to the global minimum and a lower loss function is obtained and the accuracy of the model is higher.



graph 1- from MLH course lectures,  
<https://adventuresinmachinelearning.com/stochastic-gradient-descent/>

Mini-batch gradient descent versus the rest

*In the code, we were asked to apply the model again with 32 batch compared to 64. In this case we got that the loss function is higher and the accuracy is lower since smaller batch size is more similar to the SGD update algorithm, perform the update more times as we explained.*

#### **Task 4: "Build the new \_a\_ model again and add batch normalization layers. How does it impact your results?"**

Batch normalization is performed in order to maintain a constant distribution of the inputs to the same layer, so it does not change between each update step and the layer won't have to adapt to different inputs distribution each time. After normalization, all inputs will be in the range of [0,1]. Therefore, by using it, the learning rate will be faster, the convergence to the minimum of the loss function is faster and less epochs needed. We conclude that the results will not change following the use of Batch normalization, only the calculation time will change, and this can also be seen from the results we got, the value of the loss function has not changed.

## PART 2: Convolutional Neural Network (CNN)

**Task 1:** "Have a look at the model below and answer the following:"

- How many layers does it have? 8 layers in total- 5 convolution layers(including processing operations,2 hidden fully connected layers and 1 fully connected layer of four neurons to classification.
- How many filters in each layer? There are filters only in convolution layers, in this order: 64,128,128,256,256.
- Would the number of parameters be similar to a fully connected NN?

No, the number of parameters in CNN model will be smaller than the number of parameters in fully connected NN model because in CNN activating the convolution layers with the filters reduces the number of input parameters. Therefore, less neurons for learn are required, i.e., the number of parameters that need to be learned in the CNN fully connected layers is lower according to the following formula:

$$\text{Current layer neurons } c \cdot \text{Previous layer neurons } p + c$$

Where p is lower in CNN model and therefore there are less parameters to learn.

- Is this specific NN performing regularization? Yes, it is regularization kernel in the convolution layers, dropout, and batch normalization (this reserve constant range of the inputs).