

# Machine Learning in Healthcare - 336546

## HW4 - Theoretical Questions

*Ariella Chauvart - 337917850*

*Aurelia Sebbane - 336334479*

### **Part 1**

#### Task 1

We elaborated a neural network with two hidden fully connected layers with 300, 150 neurons and 4 neurons for classification. We used the ReLU activation function for the hidden layers and He\_normal for initialization. We then compiled the model, trained it on the training set and evaluated it on the testing set.

We obtained the following results:

Test loss: 0.79

Test accuracy: 68.57 %

#### Task 2

*Change the activation functions to LeakyRelu or tanh or sigmoid. Explain how it can affect the model.*



We changed the activation function from relu to sigmoid. The results we got:

Test loss: 1.06

Test accuracy: 56%

The sigmoid function gave a lower accuracy and a higher loss than the ReLU function. The performance of the model is lower.

The ReLU is the most used activation function mostly because it is faster to compute, since it just needs to pick  $\max(0, x)$  and not perform expansive operations as in sigmoid.

The sigmoid function saturates for a very large value and for very small value, and when saturation happens, the learning algorithm has difficulties to continue learning because the derivative of the activation function will be zero causing the learning to be less efficient.  

#### Task 3

*Train the new model using 25 and 40 epochs. What difference does it makes in term of performance?*

We trained the new model using different numbers of epochs and obtained the following results:

With 25 epochs- Test loss: 1.06, Test accuracy: 56%

With 40 epochs- Test loss: 0.89, Test accuracy: 64%.

An epoch is one learning cycle where the learner sees the whole training dataset. In general, one should set the number of epochs as high as possible and terminate training based on the error rates in order to avoid overfitting.

If the number of epochs is too high meaning running and training the dataset too many times, the model may overfit.

There is a trade-off that the number of epochs influences, between improving the NN's performance and overfitting.

In our case, raising the number of epochs lead to a reduction of the test loss and augmentation of the test accuracy, meaning the model in not yet overfitted and the performance is better.

#### Task 4

*What are the advantages of the mini-batch vs. SGD?*

In this task, we ran the model with a batch size of 32 instead of 64. We obtained the following results:

Test loss: 0.87

Test accuracy: 65.14%

SGD (Stochastic Gradient descent) computes the gradient of the cost function for each training example, while mini batch gradient descent computes the gradient of the cost function for  $p$  training examples. With SGC, however the time to operate is low (it operates quickly), we receive a bad accuracy of parameter update. Mini batch operates quickly and with a high accuracy, and accurately updates the parameters, which often make it more efficient.

#### Task 5

*Build the new\_a\_model again and add batch normalization layers. How does it impact your results?*

In this task, we added batch-normalization layers and got the following results:

Test loss: 0.86

Test accuracy: 67.43%

Batch normalization is a technique for training very deep neural networks that normalizes the contributions to a layer for every mini-batch, which means mean activation output 0 and unit standard deviation. This has the impact of settling the learning process and drastically decreasing the number of training epochs required to train deep neural networks (NN).

So the model is regularized and normalized, and the learning process is quicker, the training too since the batch normalization needs less information to perform. Moreover, weight initialization is less significant.

## Part 2

### Task 1

- *How many layers does the model have?*

The model has 8 layers: 5 convolution layers, 2 hidden fully connected (FC) layers and the output layer.

- *How many filters in each layer?*

In convolution layers (CL):

1<sup>st</sup> CL: 64 filters

2<sup>nd</sup> CL: 128 filters

3<sup>rd</sup> CL: 128 filters

4<sup>th</sup> CL: 256 filters

5<sup>th</sup> CL: 256 filters

In fully connected layers (FC):

1<sup>st</sup> FC: 512 filters

2<sup>nd</sup> FC: 128 filters

3<sup>rd</sup> FC (output layer): 4 filters



- *Would the number of parameters be similar to a fully connected NN?*

In a fully connected NN, each neuron connects to every other neuron so there will be more parameters in the fully connected NN than in a convolution NN.

- *Is this specific NN performing regularization?*

Yes, this NN is performing L2 regularization at the convolution layers by using dropout, pooling and batch normalization.

### Task 2

*Now train the model with the number of filters reduced by half. What were the results ?*

We were stuck with an error at the end so we couldn't run the last task, but we would expect the accuracy of the bigger net to be the higher.