Students: Diana Salim & Kim Lif

Assignment 7 - Group 9

DIT852: Introduction to Data Science

1) Explain the data pre-processing high-lighted

Firstly, the variables x_train and x_test are being transformed into the datatype float which is specified with 32 bits. These two variables are then being normalized for each of its pixels, this is done by dividing the input variables with 255. Lastly, two y variables is a matrix of a binary sort, it gives the value of either 0 or 1 and represents the labelled pictures being categorized from the ten number classes.

However, categorical data cannot be accessed directly by machine learning algorithms as they must be transformed to numerical data. Therefore, one-hot encoding is being used in order to make it easier to manipulate numerical data. Also, a python code for normalization of the data are executed to handle the problems of values in different ranges in the model. For example, it is difficult to be able to find patterns in the data if there is a high difference in the distribution of the values.

2) a)

- Input layer in this notebook; $28 \cdot 28 = 784$ neurons. Hidden layers = 64 neurons, output layer = 10 neurons.
- The activation function, Rectified linear unit ("ReLU") is used to set negative values (x<0) to zero and if the x-value is bigger than 0 it will have its initial value. This in order to attain a better performance. On the other hand, SoftMax is used on multi-class tasks in order to classify the output based on expected outcomes. SoftMax gives an output of different values between zero and one. It will give the probability of each class outcome whereas one will choose the highest probability of a class as its target class.

$$ReLU = f(x) = max[0, x]$$
$$SoftMax = \frac{e^x}{\Sigma e^x}$$

- The total parameters of the network is 55,050.
- The input is structured in a way that the pictures in this case are on a 28 by 28 grid and the output is 10 as we have a multi-class classifier with 10 classes (see python code). The output is stored in a vector of the following dimensions: [0010000000] representing the classification results. The input layer is dependent on image size N times N creating an input matrix of the given dimensions.

b) The loss function is a categorical crossentropy that is especially used with multi-class classification and counts the difference between two probability distributions. The mathematical expression of the loss function is,

$$- \sum_{i=1}^{output\ size} y_i \cdot log\, \widehat{y_i}$$

whereas, i is an event and y_i is the probability of that event happening. The the sum of all the probabilities of the different events are equal to 1 and the minus symbol displays the loss, which will get smaller as the two probability distributions get closer. The value that is being computed with the loss function explains how well a particular set of weights perform.

This loss function is good for classification tasks and requires softmax activation on the output to work. The reason behind this is that the loss function is the sum of the log softmax output times the target output. Softmax gives a probability of classifying the target classes while the target output through hot-one encoding is a value of either 0 or 1. As a result the loss will be higher when the probability of classifying the data correctly is decreased.

In the case of predicting one picture:

*Log SoftMax probability*1 (correct class) + Log SoftMax probability*0 (incorrect class)*

This means that the loss function in this simple case will be the log of the softmax probability if the probability of classifying a picture as an 8 is 80 percent which sets the loss to be log(0.8).

c)

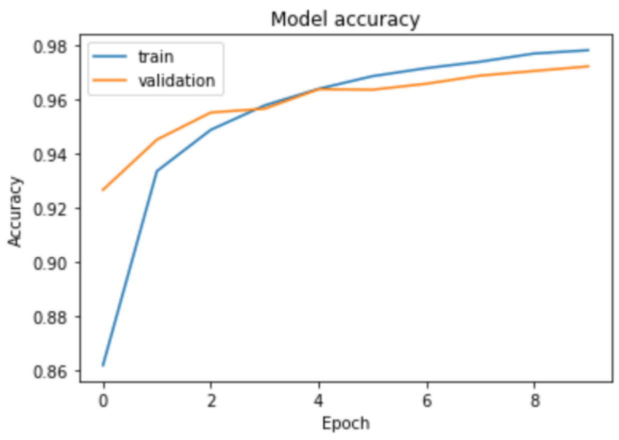

Figure 1: Accuracy for each epoch

On the figure above, it is displayed the training and validation accuracy for the ten epochs.

d)

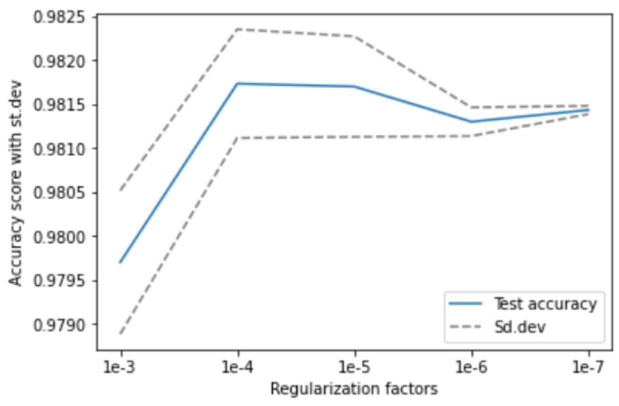

Figure 2: Validation accuracy as a function of regularization factors

Displayed on Figure 2, the final validation accuracy with standard deviation as a function of the five regularization factors are plotted. As mentioned in the question, Geoff Hinton argues that this network could get a validation accuracy of 0.9847. However, the closest result that was received to Hinton's was 0.9826 and the value furthest away from his was shown to be at 0.9786 in this test. In the case where we do not use Hinton's we get a result of 0.9822.

As a conclusion, we do not reach Hinton's accuracy which can be explained through the fact that we did not change or add any layer to reduce the risks of over- and underfitting. This, as well as, making adjustments to the hyperparameters or adding new ones. Other aspects that contribute to the difference in the result from Hinton's could be the usage of weights when adding the regularizers in the model layers. Depending on the type of weights used, it could be assumed that the results will perhaps differ and create variation in the accuracy. With some slight changes the results could match Hinton’s or perhaps, even surpass it when it comes to the accuracy of the model.