DIT852: Introduction to Data Science

Assignment 7

by

- Christoffer Wikner (931012)
- Erik Rosvall (960523)
- Group 8

1

In this code block:
you set the data type in the matrix as a float32. A float who is a 32-bit point number. 32-bits are the most common use when training a neural network. 32-bits do fit the best in RAM, so the best thing is to convert it to 32-bit immediately.

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
```

Dividing x_train and x_test with 255. This function is about normalization of the data as dividing and testing was done when loading in the data. It's best practice to normalize the data used to get the mean close to 0. Normalization of data as a generalization speeds up the learning.

```
x_train /= 255
x_test /= 255
```

Last rows in the pre-processing part:

```
y_train = keras.utils.to_categorical(lbl_train, num_classes)
y_test = keras.utils.to_categorical(lbl_test, num_classes)
```

to_categorical: this function converts a vector of type class to a binary matrix.

num_classes: number of classes that are passed in, classes in this case stands for the number 0-9.

One-hot encoding is a commonly used method when it comes to machine learning. This method is often used to address categorical data. In almost every machine learning

algorithm the input must be a numerical value, in other words variables that are attached to categorical data must be converted during the pre-processing stage.

2

A)

In this code we were provided, the activation functions that was given was relu. Relu stands for rectified linear activation function. Relu is a part of the linear function and it's constructed that it always will give a direct answer. If the answer is positive, if its negative, it will just return a 0. Relu is considered as a default these days and are compatible with a wide range of neural networks since it makes it easy to train which makes it easy to get higher performance.

ReLU is a function, it is defined as following:

$$f(x) = max[0, x]$$

The softmax function is an activation function used most often in the final output layer of an NN model. Most commonly in NN models that use cross-entropy. It classifies the actual output based on its exception. What this means is that it will predict the probability of each individual class as an outcome and will choose the one with the highest probability of these.

Softmax is defined as:

$$\frac{e^{x}}{\Sigma e^{x}}$$

Parameters of the network is in total 55.050

This Neural Network has one input layer, two hidden layers and one output layer.

Number of neurons

Each picture are 28x28 pixels

$$28 * 28 = 784$$

Layers:

- Flattened vector ("input layer"): 784
- Hidden layer 1 (input layer): 64

- Hidden layer 2 (output layer): 64
- “Output layer”: 10

784 – 64 – 64 -10.

The reason there are 10 in the output layer is that in hyperparameters num_classes = 10

B)

The NN is using *categorical_crossentropy* as a loss function. This cross entropy is most commonly used in NN when it comes to multi-class classification.

it can be found in this code block

```
loss=keras.losses.categorical_crossentropy
```

Mathematical description:

$$Loss = -\sum_{i=1}^{outputsize} y_i * log\, \hat{y}_i$$

Categorical cross entropy is a good technique for classifications problems, like the MNIST problem especially in this instance where the model makes use of the softmax function which is a prerequisite for working. Softmax as previously discussed will give a probability and makes use of one-hot encoding and classifies it to either a value of 0 or 1. Based on these variables the loss will be higher when it deems that the probability of classification happening correctly is lower. What this looks like is that Softmax p*1 for corect class and p*0 for incorrect class.

C)

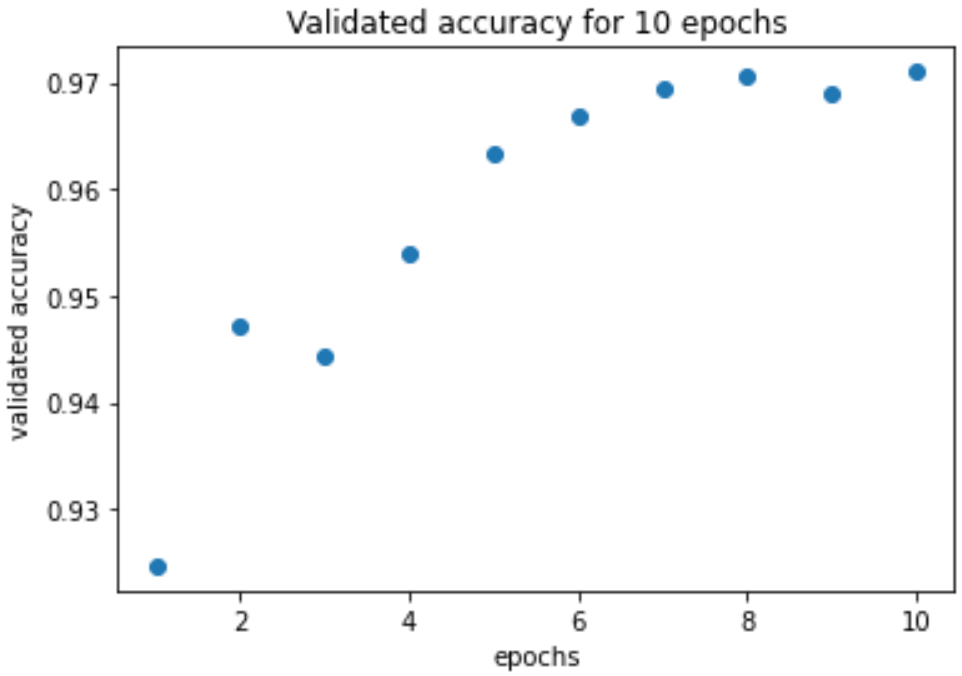

Fig1: Validated accuracy for 10 epochs

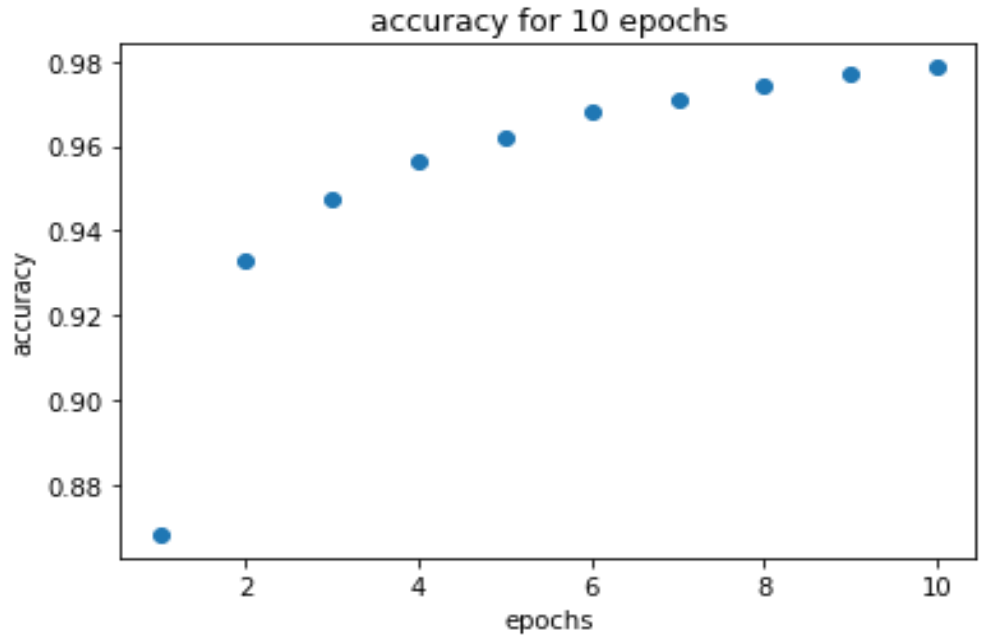

Fig2: Accuracy for 10 epochs

D)

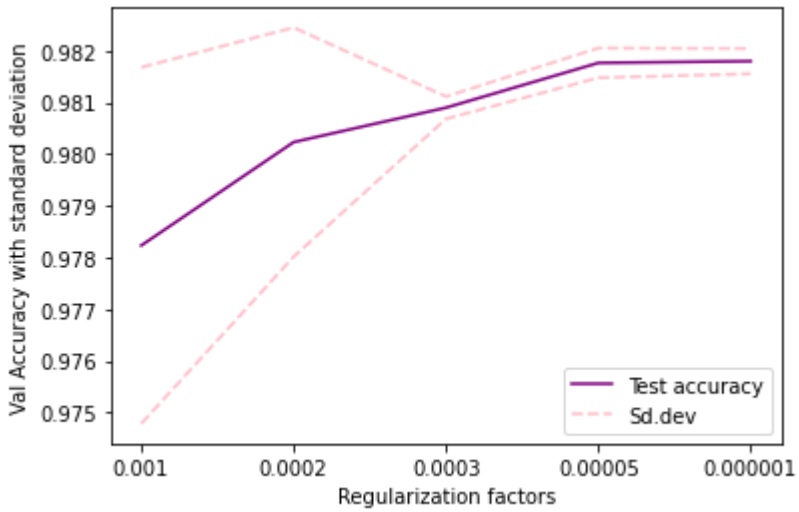

Fig3: Validated accuracy as a function of regularization factors

Displayed in fig 3 is the validation accuracy with standard deviation plotted against and as a function of regularization factors. Hinton argues that 0.9847 is the validated accuracy this NN can achieve. The highest value we could achieve was 0.9833 which is a bit off Hinton' result but still reasonably close.

Things that could account for the lower validated accuracy compared to Hinton's result of 0.9847. A high probability that Hinton has optimized the batch sizes, epochs, amount of layers and the amount of units in each layer. Hinton could also have used momentum (initial momentum, final momentum) instead of gradient descent, there is also a possibility that Hinton has used a static weight to the Neural Network.