Assignment 7

by

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

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 divides the training and testing. To get an unbiased evaluation of results and assess the predictive power of the machine learning model we must evaluate using new data. We reach this by splitting the dataset into training and testing. You cannot validate a model by using the data it is trained on.

```
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.

lbl: This turns a 'hard' class of label assignments to a 'soft' label assignment. Since a 'hard' label are binary. In other words, either it has a label or not. By converting it to a soft label, it becomes a score. This score can be compared with a likelihood or probability.

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 categorial 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.

This Neural Network has on 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)

They are using *categorical_crossentropy* as a loss function. In other words, the values **mean**.

it can be found in this code block

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

Mathematical description:

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

This loss measure the probability distribution between the two values. The minus ensures that the loss gets smaller if/when the distribution gets closer to each other. The two values stand for low probability event and high/higher probability event. Cross-entropy is the difference from a provided random value for a specific set of events. Entropy is a specific number represented in bits that are supporting/helping one of the specified sets that are derived from a distribution of probability. There are two states for entropy distribution, distort which provides a low entropy. When the probability is equal (or close to equal) the entropy is higher.

Yi is the probability for how many times I occurs, the sum of all y_i is 1. This states the there are only one event that is happening. The – sign works as a safeguard. It keeps the provided loss to a minimum when all distributions are getting closer. $\hat{y}_i$ represent the scalar value (i-th) that the model is outputting. Log is used to simplify long and complicated calculations. By taking the log of $\hat{y}_i$ the scalar value is much easier to read.

Categorical crossentropy is a good technique for classifications problems, like the MNIST problem. categorical crossentropy is constructed to give a high probability for the correct digit, hence a lower probability for the non-correct digit.

c)

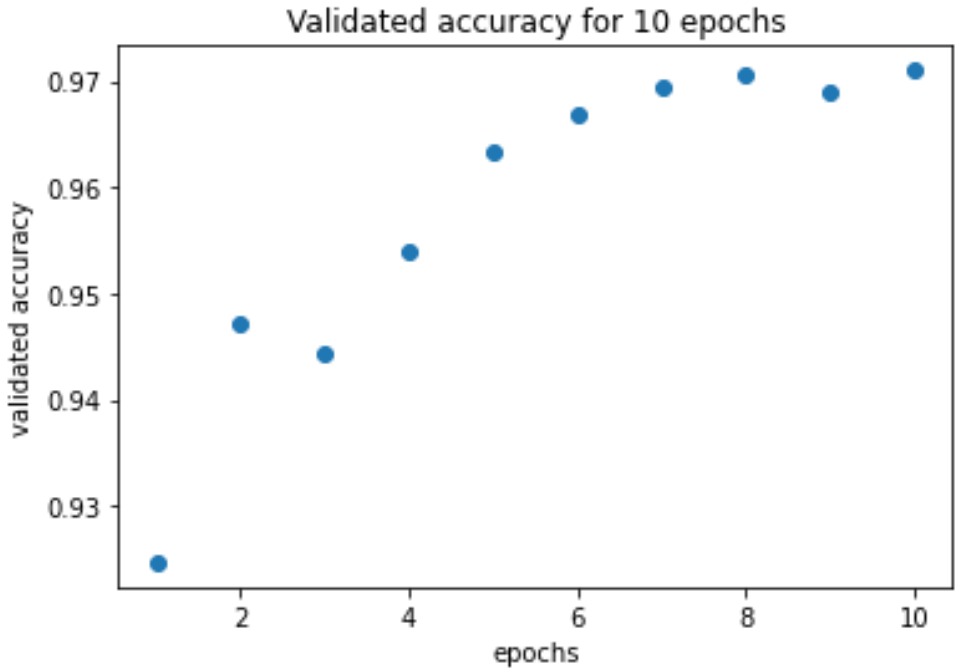


Fig1: Validated accuracy for 10 epochs

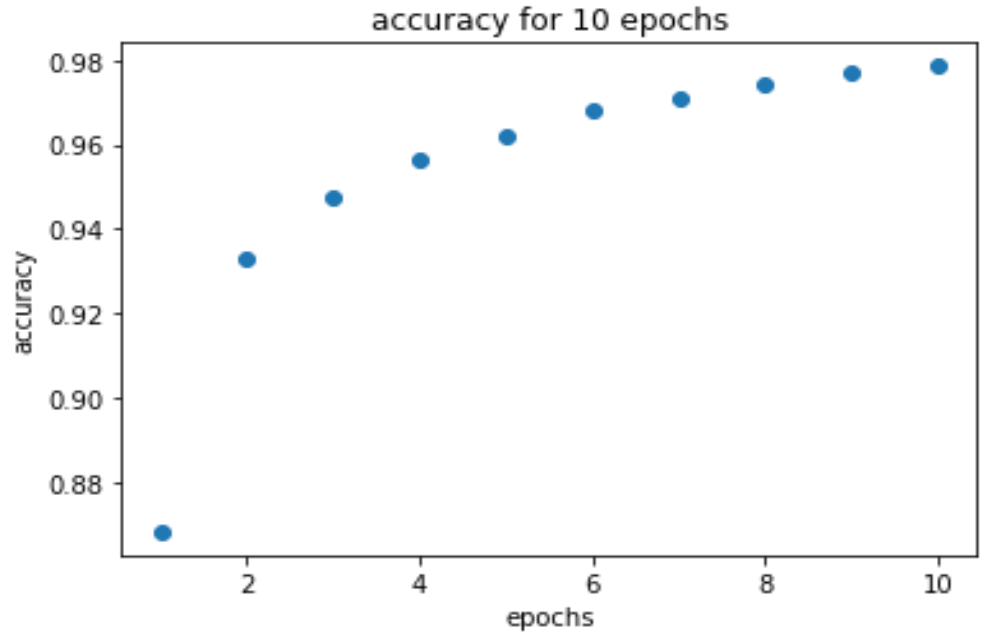


Fig2: Accuracy for 10 epochs

D)

PART 1

Running a 3 layer neural net with 500 and 300 hidden layers units respectively trained for 40 epochs we reached a validated accuracy of 0.9829999804496765.

PART 2

Max values from all runs

max 0.001:

- 0.9196000099182129
- 0.9179999828338623
- 0.9200999736785889

max : 0.005

- 0.9567000269889832
- 0.9564999938011169
- 0.9560999870300293

max: 0.008:

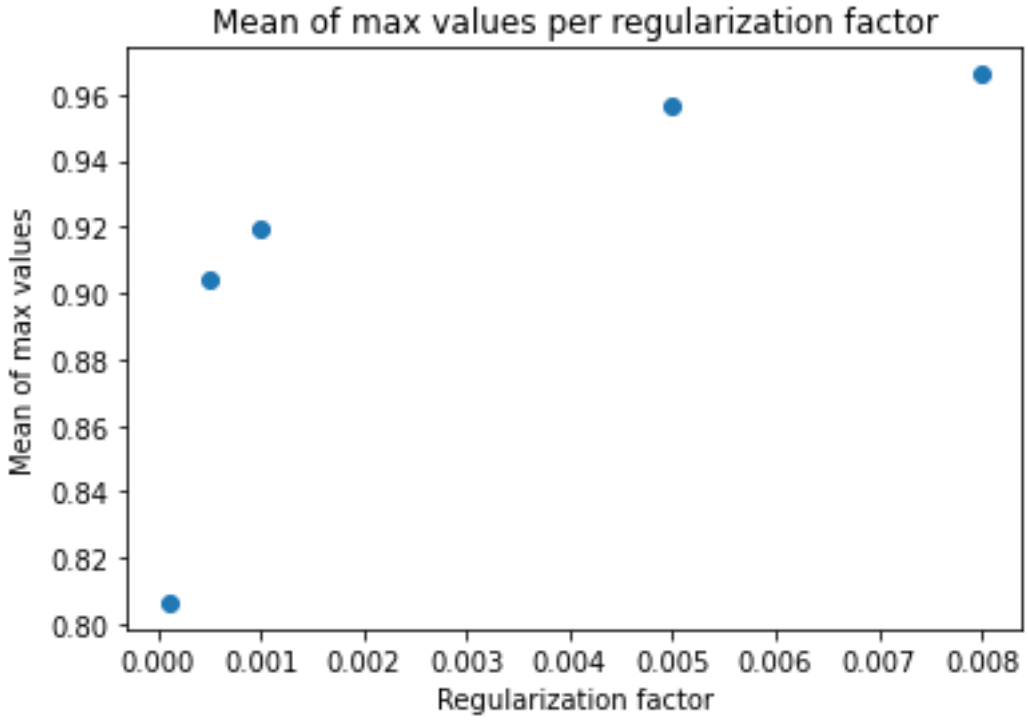
- 0.9657999873161316
- 0.9666000008583069
- 0.9656999707221985

max 0.0001:

- 0.8055999875068665
- 0.8263999819755554
- 0.7876999974250793

max 0.0005:

- 0.9039000272750854
- 0.9043999910354614
- 0.9035000205039978



Standard deviation from all runs

deviation 0.001:

- 0.06309258302592037
- 0.0656045393804049
- 0.08142213493982044

deviation 0.005:

- 0.02377655110530879
- 0.025387090346489814
- 0.025376603549890037

deviation 0.008:

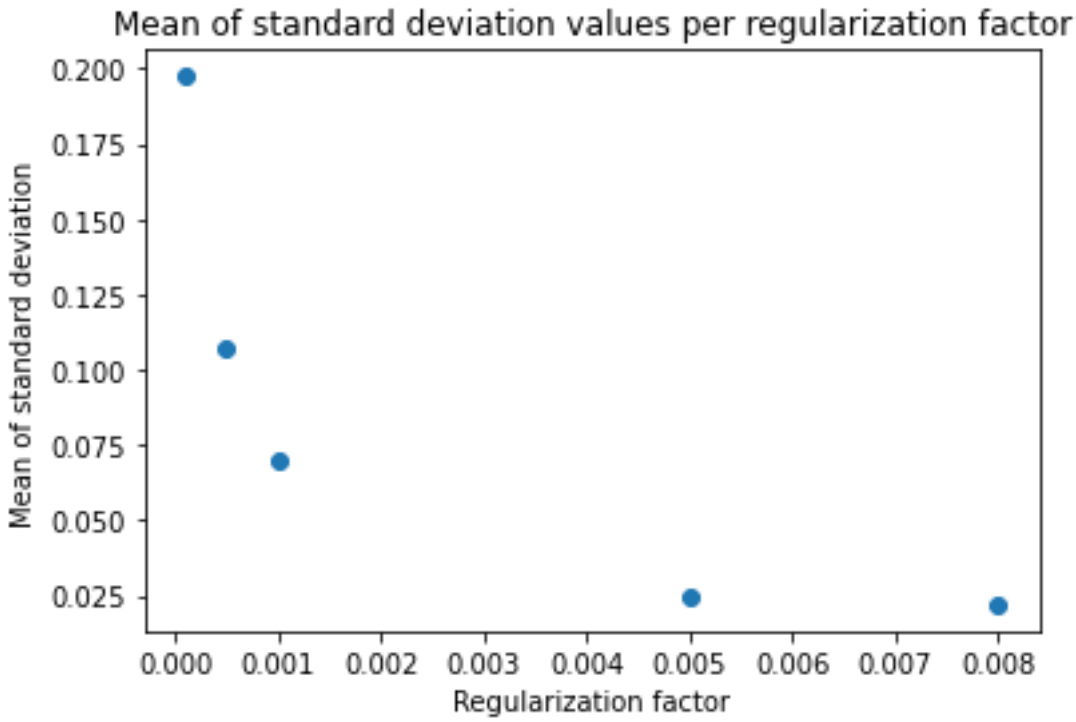
- 0.022381706799999467
- 0.021606646281912313
- 0.022289511042831656

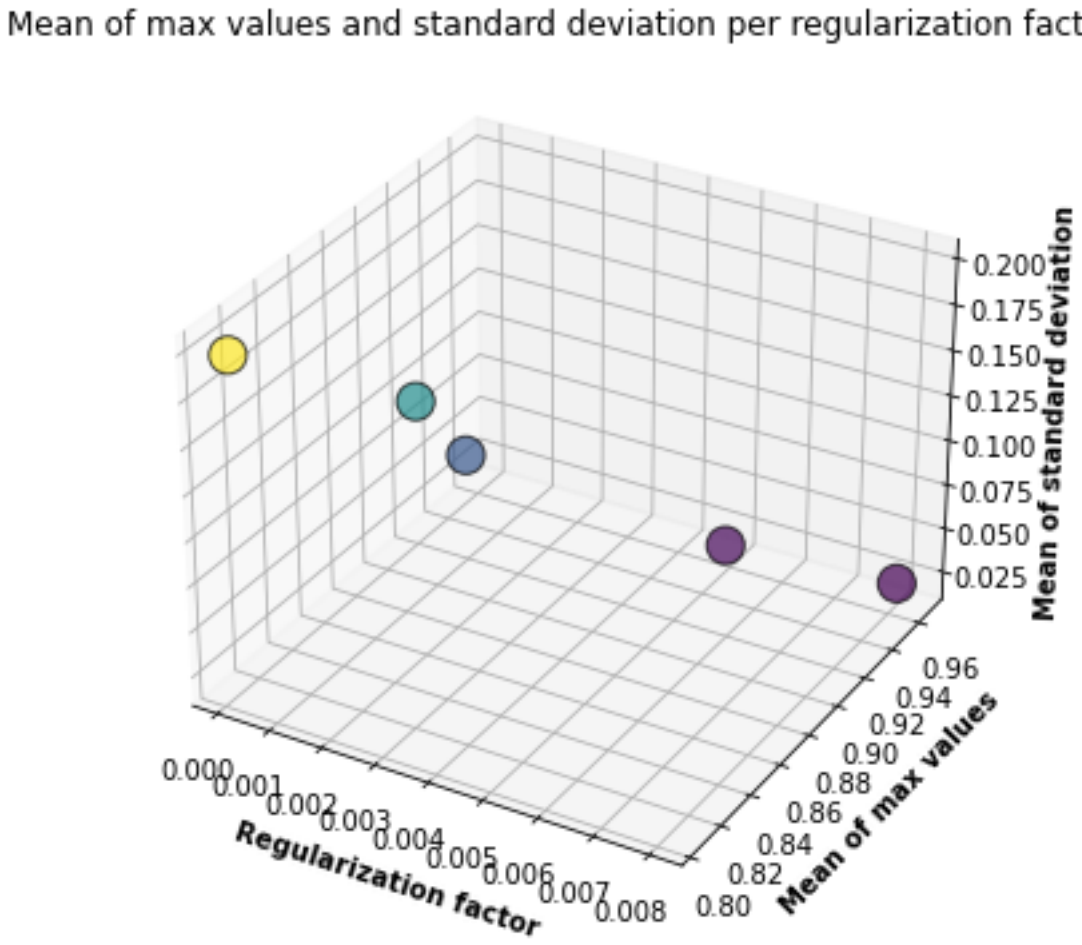
deviation 0.0001:

- 0.20507056766918563
- 0.20331022052862852
- 0.18407025558993959

deviation 0.0005:

- 0.10300974946673583
- 0.11080766196883153
- 0.1083929162747305





PART 3

The closest we got was with a regularization factor of 0.1 for 40m epochs, with 2 hidden layer (500 and 300). The result was 0.9829999804496765.

With our tests of 5 regularization factors between 0.001 and 0.0005, the highest max result we reached was 0.9666000008583069 when regularization factor was 0.008.

Things that could account for the lower validated accuracy compared to Hinton's result of 0.9847. The computer hardware is one factor. Thought here is a higher 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 have used a static weight to the Neural Network.