Deep Learning Summer Course - Exercise 4

Summer 2022
Monday 15-8-2022

To do these exercises, you will use Python with 3 and the following packages:

- TensorFlow. The used backend used for Keras
- Keras. Good front-end package for easier integration of neural networks
- Matplotlib. This package allows you to graph your data.
- Seaborn. This extends MatplotLib with new plots.

You are not strictly forced to use these packages, but it is highly recommended. Feel free to use other packages you think are necessary.
We will go through your results of this exercise on Tuesday 16-8-2022, so you are expected to have completed them.
Question or comments: Tobias Greisager Rehfeldt

This exercise will extend on the model created during Exercise 1. Today you need to create a convolutional network that detect the numbers in the images Load the dataset by:

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
import seaborn as sns
import matplotlib.pyplot as plt
# Loads the data
(train_data, train_labels), (test_data, test_labels) = mnist.load_data()
# Plots a single digit from the data
sns.heatmap(train_data[1, :, :])
plt.show()
# one-hot encoding the labels to work with the output layer
num_classes = 10
train_labels = to_categorical(train_labels, num_classes)
test_labels = to_categorical(test_labels, num_classes)
```

1. Look at the data, do you have to do anything before you feed it to your network? (hint: reshape and normalize data)

2. After you've made your CNN and trained it, try playing around with it. Can you improve the accuracy?

3. Plot some incorrectly labelled data points to see if it makes sense it was wrongly classified. Then visualize one of the following (Feel free to do more than one!):
    a. The first convolutional layers’ filters. Do this before and after training and comment on the filters

```
kernels = model.get_layer(name='layer_name').get_weights()[0][:, :, 0, :]
```

    b. The feature maps of the first convolutional layer (see slide 82)
    c. The input maximization for any convolutional layer (see cnnvis.py)