INF201 Exercise 6

Fill out group member info and NMBU-emails.. Only **one member has to upload a .ipynb and .pdf file to Canvas**.

name_1: Mehrad Farmahini Farahani
name_2: Eskil Digernes
nmbu_email_1: mehrad.farmahini.farahani@nmbu.no
nmbu_email_2: eskil.digernes@nmbu.no

In [ ]:
```python
import numpy as np
from torchvision import datasets, transforms

""" 1. Restructure exercise 5 using classes"""
# Class "Layer" which stores the weights and biases
class Layer:

    def __init__(self, n_input, n_output):
        self.weights = np.random.rand(n_output, n_input)
        self.biases = np.random.rand(n_output)

    def sigma(self, x):
        y = np.dot(self.weights, x) + self.biases
        return np.maximum(y, 0)


# Class "Network" which stores a list of layers
class Network:
    def __init__(self):
        self.layers = []

    def add_layer(self, layer):
        self.layers.append(layer)

    def feedforward(self, x):
        for layer in self.layers:
            x = layer.sigma(x)
        return x


    """ 2. "Read" the weights and biases from the .txt files """
    def read(self, file_prefix='', file_extension='txt'):
        for i, layer in enumerate(self.layers, start=1):
            weight_file = f"{file_prefix}W_{i}.{file_extension}"
            bias_file = f"{file_prefix}b_{i}.{file_extension}"

            layer.weights = np.loadtxt(weight_file)
            layer.biases = np.loadtxt(bias_file)


    """ 3. "Evaluate" function in the Network class from a given input x """
    def evaluate(self, x):
        output = self.feedforward(x)
        return output

# Getting the MNIST dataset and images
def get_mnist():
    return datasets.MNIST(root='./data', train=True, transform=transforms.ToTensor(), download=True)

def return_image(image_index, mnist_dataset):
    image, label = mnist_dataset[image_index]
    image_matrix = image[0].detach().numpy()  # Grayscale image, so we select the first channel
    return image_matrix.reshape(image_matrix.size), image_matrix, label



# Create a network instance and add layers
neural_network = Network()
n = [784, 512, 256, 10]

# Adding layers with given dimensions
for i in range(len(n)-1):
    layer = Layer(n[i], n[i+1])
    neural_network.add_layer(layer)


# Read the weights and biases from the files
neural_network.read()
```

```python
#  Load the MNIST dataset
mnist_dataset = get_mnist()



""" 4. Evaluate the neural network at different images"""
image_list = [ 19962, 19963, 19964, 19965, 19966, 19967] # Selecting random images?

# Evaluation of the results. Comment out the print statements to see the results
for image in image_list:
    x, image_matrix, label = return_image(image, mnist_dataset)
    output = neural_network.evaluate(x)
    nn_response = np.argmax(output)
    # print(f"Image {image} shows the number {label}")
    # print(f"The neural network response of image {image} is {nn_response} ")



""" 5. Print out the response to image 19961 """
image_index = 19961
x, image_matrix, label = return_image(image_index, mnist_dataset)
output = neural_network.evaluate(x)

# Determine the predicted class based on the highest output value
nn_response = np.argmax(output)

# Print the results
print(f"Image {image_index} shows the number {label}")
print(f"The neural network's response of image {image_index} is {nn_response} ")

# Determine the predicted class based on the highest output value
nn_response = np.argmax(output)
```

```
Image 19961 shows the number 4
The neural network's response of image 19961 is 4
```