Assignment 2
E-mail a pdf with your solutions *and* a zip-file with your python scripts to a.schoenhuth@cwi.nl before November 27, 2019, 3 pm. At the lecture, provide also a printout of the solutions (but no longer the code!). In your email include your 7-digit UU username. Include a python script for each sub question (except exercise 1a), so your zip file will eventually contain the following scripts: exercise1b.py, exercise1c.py, exercise1d.py, exercise1e.py, exercise2a_network1.py, exercise2a_network2.py, exercise2a_network3.py and exercise2b.py.

In this assignment you will adapt the script tutorial.py to build a convolution neural network.

Throughout the exercise you will need the help of explanations by others on the internet. Of course we could have provided you with explanations of all the functions you need to use, but by figuring out things yourself instead you will learn how to implement features in tensorflow that you haven't seen before, which makes you an independent user of tensorflow for your future projects.

Throughout the exercises use (only when applicable!) 50 epochs, a batch size of 100, 5 fold cross validation and gradient descent optimization with a learning rate of 0.05 (note that some of these are different from assignment 1!) Use cross entropy as a loss function with softmax activation in the output layer. Use your 7-digit UU username as the random seed for both cross validation and the variable initializers.

Training a convolutional network can take a long time. Therefore, whenever you use cross validation, it suffices for this assignment to train and validate only for one of the five folds. You can do so by adding a "break" at the end of the for loop. This looks like this:

```
for train_idx, val_idx in kf.split(x_train):
    [code for cross validation training and validation]
    val_losses.append(loss_val)
    break
```

Exercise 1

a) While we needed to reshape the images into a vector for the fully connected layers, convolution layers require a different shape of the tensor. Replace

```
xdata = tf.reshape(X, [-1,imsize[0]*imsize[1]])
```

by

```
xdata = tf.reshape(X, [-1,imsize[0],imsize[1],1])
```

Why do you need to do this? In your answer explain:

- why the input data was reshaped into a vector in the fully connected neural network of assignment 1, but is not reshaped into a vector for the convolution neural network;
- what the last dimension of "1" means.

[2 points]

b) Build the following network:
- input layer;
- a convolution layer with a 5x5 filter and 6 output channels, stride 1, padding and followed by activation function relu (use tf.nn.conv2d for the convolution layer);
- a max pooling layer with a 2x2 filter and stride 2x2, zero padding;

- a fully connected layer with 128 nodes and sigmoid as activation function;
- output layer.
[9 points]

c) Consider the network of exercise b). What is the size of the tensor that is the output of the convolution layer? What is the size of the tensor that is the output of the pooling layer? Explain why. *[3 points]*

d) Change the padding of the convolution layer to zero padding. What is now the shape of the output of the convolution layer? Explain why. *[1 point]*

e) Change the stride of the convolution layer to 2x2 (keep padding as in exercise b). What is now the shape of the output of the convolution layer? Explain why. *[1 point]*

Exercise 2

a) Optimize the network architecture by finding the best among the list of architectures below. Report the loss values for each of the architectures, choose the best model (and motivate your decision) and report its final performance (indicate which metric you used). *[5 points]*

Network 1
- input layer;
- a convolution layer with a 5x5 filter and 16 output channels, stride 1, padding and followed by activation function relu (use tf.nn.conv2d for the convolution layer);
- a max pooling layer with a 2x2 filter, stride 2 and zero padding;
- a convolution layer with a 3x3 filter and 32 output channels, stride 1, padding and followed by activation function relu (use tf.nn.conv2d for the convolution layer);
- a max pooling layer with a 2x2 filter, stride 2 and zero padding;
- a fully connected layer with 128 nodes and sigmoid as activation function;
- output layer.

Network 2
- input layer;
- a convolution layer with a 5x5 filter and 16 output channels, stride 1, padding and followed by activation function relu (use tf.nn.conv2d for the convolution layer);
- a max pooling layer with a 2x2 filter, stride 2 and zero padding;
- a convolution layer with a 3x3 filter and 32 output channels, stride 1, padding and followed by activation function relu (use tf.nn.conv2d for the convolution layer);
- a max pooling layer with a 2x2 filter, stride 2 and zero padding;
- a fully connected layer with 256 nodes and sigmoid as activation function;
- a fully connected layer with 128 nodes and sigmoid as activation function;
- output layer.

Network 3
- input layer;
- a convolution layer with a 7x7 filter and 16 output channels, stride 1, padding and followed by activation function relu (use tf.nn.conv2d for the convolution layer);
- a max pooling layer with a 2x2 filter, stride 2x2 and zero padding;
- a convolution layer with a 5x5 filter and 32 output channels, stride 1, padding and followed by activation function relu (use tf.nn.conv2d for the convolution layer);
- a max pooling layer with a 2x2 filter, stride 2x2 and zero padding;
- a fully connected layer with 256 nodes and sigmoid as activation function;
- a fully connected layer with 128 nodes and sigmoid as activation function;
- output layer.

b) Build a network of your choice with at least 2 convolution layers, each followed by a pooling layer, and at least one fully connected layer at the end. Describe the architecture in the same way as was done in exercise b). Train your network and provide its performance. Did you use train and test data, or did you use train, validation and test data? Explain why. *[6 points]*