Assignment : Image Segmentation
Signal and Image Processing

March 16, 2023

You can work on this assignment and submit your solution (report and code) as a GROUP. This assignment counts towards your grade and have to be submitted in order to pass the course. You must follow the report guidelines found in `guidelines.pdf`. For test images, scripts and data files - see `Files/Material/Test Images/Week 7` on Absalon. The page limit for this assignment is **10 pages** including everything, i.e. illustrations and code snippets.

1 The Hough Transform

1. (1 point) Implement a straight line Hough Transform - do not use a library function that computes the Hough transform such as the `hough_line` method in `scikit-image`.
   Deliverables: Include essential code snippets illustrating your implementation. Describe how your implementation works and possible issues your implementation can have (e.g. computational complexity).

2. (1 point) Test your implementation on the image `cross.png`.
   Deliverables: Plot both the Hough transform of the image and the detected lines overlayed on `cross.png`. Compare the output of your implementation to the output of the `hough_line` method in `scikit-image`.

3. (1 point) Use the `hough_circle` method of `scikit-image` to make a segmentation method that can segment the coins in `coins.png`. You will likely need to apply `hough_circle` to the edges in `coins.png` instead of the actual image, e.g. run Canny edge detection from `scikit-image` and feed the output to the circle Hough transform.
   Deliverables: Describe the method and visualize the results.

2 Machine learning based segmentation

Here, we will employ a machine learning based method to segment 2D brain images. The classifier will be a 2D neural network based on the 3D segmentation work of Yuan-Ching Teng and Akshay Pai described in Pai et al. "Characterisation of errors in deep learning-based brain MRI segmentation".

Download the tensorflow library – e.g. with `conda install tensorflow` or `pip install tensorflow-cpu` – and the script `keras.py` from Absalon. The neural network and its design is not the focus of this assignment. Instead, you can use the script to load the pretrained network. The pretrained weights are available in the file `keras.h5` and the script loads the weights with the command

```
model.load_weights('keras.h5')
```

You are of course also welcome to train the model in the script yourself if you have sufficient computing power available (it takes around 12 hours on a standard laptop) but this is purely optional.

The test data used below is available in the file `test.npz`. If you choose to train the network yourself, the entire dataset (test+train) is in the file `train.npz`. The test data can be loaded as below:

```
test = numpy.load('test.npz')
x_test = test['x_test']
y_test = test['y_test']
```

The test images used in problem 3) and 4) below are available in the file `test_images.zip`.

The model takes 29x29 image patches and outputs labels corresponding to the class of the center pixel in a segmentation of the image. The segmentation has 135 classes.

1. (1 point) **Deliverables:** Explain the general patch-based machine learning approach to segmentation: What is train and test data? Given a patch, which output do we wish the trained algorithm to predict? What are general pros and cons compared to segmentation methods that do not use training, e.g. intensity based and edge based methods?

2. (1 point) **Deliverables:** Use Keras and the test data to evaluate and report the test accuracy. Explain what the accuracy tells you about the performance of the segmentation.
   You can read the Keras documentation at `https://keras.io/models/sequential/`, e.g. `model.evaluate`. Please note that the patches must have shape `[29,29,1]` (note the '1') as you can see by inspecting the shape of `x_test`. (Hint: the accuracy should be higher than 0.9, if not, something is wrong)

3. (1 point) Load one of the test images. Write a function to extract 29x29 patches from the image (use zero-padding at the borders).
   Deliverables: Include the code for your function in the report. Use the network to produce an image containing the predicted segmentation (the value of each pixel is a class label). Beware that the intensity values in the images/patches must be scaled to be equivalent to `x_train`. This can be done by dividing by 255: 'patch = patch / 255'.

4. (1 point) Load the ground truth segmentation for your test image (in `test_images/seg`). Write a function to evaluate the Dice coefficient against your predicted segmentation.
   Deliverables: Include the code for your function in the report. Use the function to report the mean Dice coefficient computed over all classes.

5. (1 point) See if you can improve the Dice coefficient or other aspects of the segmentation by changing the classifier, either by using a different machine learning method (K-means, SVM, etc.) or by changing the neural network topology, adding dropout, performing other modifications to the model, or improving the training.
   Deliverables: Explain what you did and report on the results.