Fall 2018: CSI5139Q
Assignment 2

Due: Tuesday, October 30th, 2018, 11:00pm in Virtual Campus
University of Ottawa - Université d'Ottawa

Jochen Lang

1 Image Labelling with Neural Networks

This assignment will give you a chance to familiarize yourself with the Keras API with tensorflow backend. We will use the Extended Outex texture dataset from the University of Oulu and LAGIS-FRE CNRS which can be downloaded from `http://lagis-vi.univ-lille1.fr/datasets/outex.html`. You are looking for an archive called `Outex_TC_00030.tar.gz`. This archive contains colour images of 68 different texture classes. There are $10,880$ testing images and $1,360$ training images. There are two text files which provide a label for each images. We will however use the test images for training and train images for testing.

You must use Keras with the tensorflow backend for this assignment, i.e., the package `tf.keras`. You may use other tensorflow packages and scikit-learn and scikit-image but *not* other deep learning framework, e.g., caffe, pytorch, theano etc.

1.1 Getting Started [1]

You will need to download the texture image and organize them suitably for tf.keras.preprocessing. In particular, have a look at `tf.keras.preprocessing.ImageDataGenerator.flow_from_X` where X is either directory or dataframe. Note that this will in turn use the python image library (PIL), or rather its active fork pillow. This time the images are bpm images. Make sure to be able to load training and test images. Your jupyter notebook must work with the images unpacked to the subdirectory `Outex-TC-00030` from your notebook.

1.2 Perceptron [2]

Build a multilayer perceptron model (similar to the MNIST example shown in class) to classify the texture into their 68 materials. You must reduce the size of the images. I suggest using an image size of 32×32. For this part of the assignment, you must build and train the Multi-layer perceptron model in Keras. The sequential model API is perfect for this kind of task. Plot a loss curve showing both training and testing results.

1.3 LeNet-5 [2]

Now use full size images with a CNN in Keras from scratch. Adapt Yann LeCun's LeNet-5 network structure (Note you will have to use different sizes in order to work with the texture image). Plot again a loss curve showing both training and testing results.

1.4 VGG-16 [3]

For this part adapt the the VGG-16 network for the task. The pre-trained network is available from `tf.keras.applications.vgg16`. You want to suitably remove some layers (importing with `include_top=False` is a good start but less layers may be sufficient) and add fully-connected layer(s) and a softmax classifier at the output. You will need to train the new layers with the weights of the existing VGG layers fixed. Once you have a working classifier for the task, try to improve the classification result by training some layers a bit more (typically the higher-level) layers.

1.5 Visualization with TSNE [Bonus]

As a bonus try to visualize different input or layers by projecting them into 2D with `sklearn.manifold.TSNE` which implements t-distributed Stochastic Neighbor Embedding by van der Maaten and Hinton. But be careful with the dimensionality of the data and sizes used for visualization. For bonus points, you must discuss the graph that you produce.

2 Submission

You will need to submit your solution in a Jupyter file, do *not* submit the data. Make sure you have run all the cells. All text must be embedded in the Jupyter file, I will not look at separately submitted text files. If your Jupyter file needs a local python file to run, please submit it as well.Assignment submission is only though Virtual Campus by the deadline. No late submissions are allowed, you can submit multiple times but only your last submission is kept and marked.