Lecture “Advanced Data Analytics”

Problem Set 5

Simon Scheidegger

<u>Exercise 1</u>

<u>Tensorflow and Keras</u>

This exercise is to familiarize yourself with Tensorflow & Keras. You should repeat exercise 1 with Tensorflow & Keras and approximate the same functions with Deep Neural Networks. This is obviously a regression setting.

Look at the test functions below. Pick **three** of those test functions. Choose the parameters *w* and *c* in meaningful ways.

a) use an DNN, and compute for those three functions the maximum and average error for increasing level, as we did it in the lecture. Choose the dimension of the function for d =2, 4,10. The errors should be computed by generating 1,000 uniformly distributed random test points from within the computational domain $[0,1]^d$.

Vary the following things to gain a feeling for DNN.
b) the network architecture---that is, the number of hidden layers, the number of neurons per layer, activation functions.
c) the amount of training data you need to generate to approximate a function accurately.
d) the number of epochs and the batch size.
e) try out the “drop-out” functionality.
f) try out the three different optimizers we proposed you in the teaching examples in class (“Adam”, etc.)
g) discuss which combinations work best for above's functions.
h) generate convergence plots (number of points versus error).

1. OSCILLATORY: $$f_1(x) = \cos\left(2\pi w_1 + \sum_{i=1}^{d} c_i x_i\right),$$
2. PRODUCT PEAK: $$f_2(x) = \prod_{i=1}^{d} \left(c_i^{-2} + (x_i - w_i)^2\right)^{-1},$$
3. CORNER PEAK: $$f_3(x) = \left(1 + \sum_{i=1}^{d} c_i x_i\right)^{-(d+1)},$$
4. GAUSSIAN: $$f_4(x) = \exp\left(-\sum_{i=1}^{d} c_i^2 t(x_i - w_i)^2\right),$$
5. CONTINUOUS: $$f_5(x) = \exp\left(-\sum_{i=1}^{d} c_i |x_i - w_i|\right),$$
6. DISCONTINUOUS: $$f_6(x) = \begin{cases} 0, & \text{if } x_1 > w_1 \text{ or } x_2 > w_2, \\ \exp\left(\sum_{i=1}^{d} c_i x_i\right), & \text{otherwise.} \end{cases}$$

<u>Exercise 2</u>

<u>Tensorflow and Keras</u>

Apply the simple Tensorflow & Keras to classify the MNIST data set (located **dat/mnist.pkl.gz**)

To do so, proceed as follows.

a) Describe in few words the MNIST data set.
b) Try out various DNN architectures to run a test classification on the data set.
Moreover, play with different 3 sizes of test / training sets.
c) Apply different optimizers and batch sizes (e.g, SGD, RMSProp, Adam,...).

For a) -c): What are the respective performances?

d) How do you measure the performance of your program?

<u>Exercise 3</u>

Review the Python notebook **dat/deep_classifiers.ipynb**. This notebook uses Keras to build three simple feedforward networks applied to the half-moon problem: a logistic regression (with no hidden layer); a feedforward network with one hidden layer; and a feedforward architecture with two hidden layers. The half-moons problem is not linearly separable in the original coordinates. However you will observe—after plotting the fitted weights and biases—that a network with many hidden neurons gives a linearly separable representation of the classification problem in the coordinates of the output from the final hidden layer.
Complete the following questions in your own words:

a.) Did we need more than one hidden layer to perfectly classify the half-moons dataset? If not, why might multiple hidden layers be useful for other datasets?
b.) Why not use a very large number of neurons since it is clear that the classification accuracy improves with more degrees of freedom?
c.) Repeat the plotting of the hyperplane, in Part 1b of the notebook, only without the ReLU function (i.e., activation="linear"). Describe qualitatively how the decision surface changes with increasing neurons. Why is a (non-linear) activation function needed? The use of figures to support your answer is expected.

<u>Exercise 4</u>

<u>Bitcoin prediction with Recurrent NNs, LSTM</u>

The file **dat/coinbase.csv** contains a time series of bitcoin prices.
The goal of this exercise is to use RNNs / LSTMs in the fashion introduced in the class predicting minute head mid-prices from minute snapshots of the USD value of Coinbase over 2018.

Exercise 5* (voluntary exercise – we will do a poll on who creates the funniest picture)

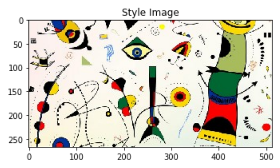
Style transfer

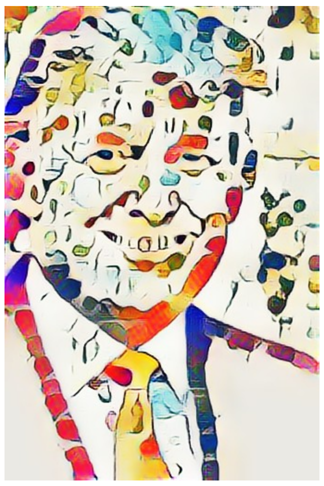
There is a tutorial on how convolutional neural networks can be used for a thing called “style transfer”, located here

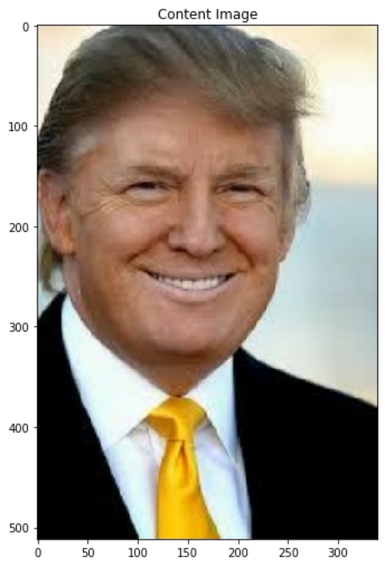
style_transfer/style_transfer.ipynb

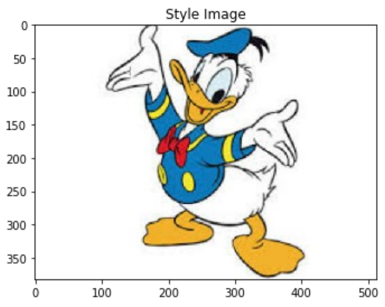
At this point in time, you are not supposed to fully understand all the details.

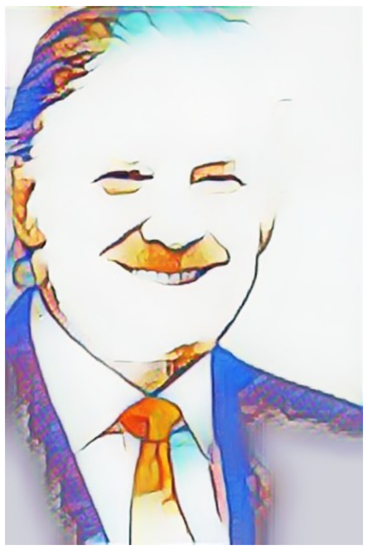
However, what you should understand is that you can mimic some figure in the style, e.g., of a painter, e.g., Miro and Donald



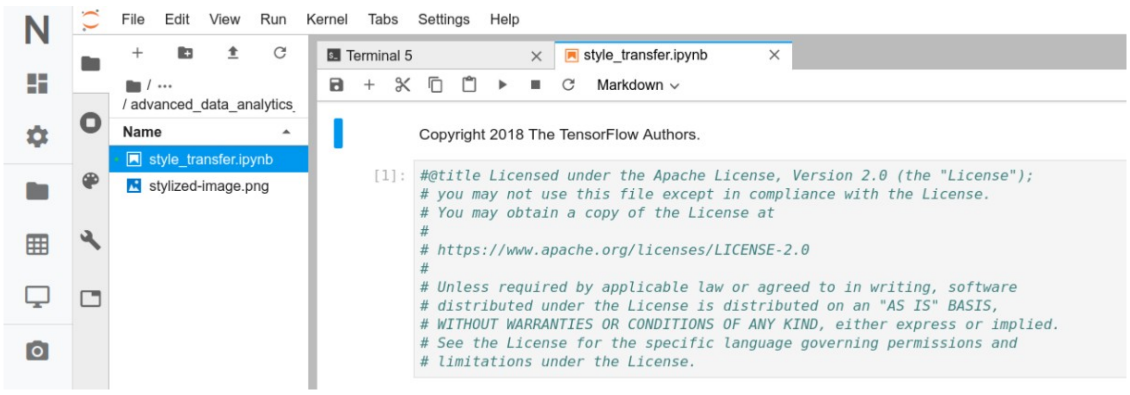








a) Work through the tutorial. To do so, open the **style_transfer** notebook:



Copyright 2018 The TensorFlow Authors.

```
[1]: #@title Licensed under the Apache License, Version 2.0 (the "License");
     # you may not use this file except in compliance with the License.
     # You may obtain a copy of the License at
     #
     # https://www.apache.org/licenses/LICENSE-2.0
     #
     # Unless required by applicable law or agreed to in writing, software
     # distributed under the License is distributed on an "AS IS" BASIS,
     # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
     # See the License for the specific language governing permissions and
     # limitations under the License.
```

b) Create a funny picture that uses style transfer, and submit it to the TA via EMAIL!

We will do an anonymous poll in the class, and the winner gets fame!