Fall 2021: ELG7186
Assignment 2

Due: Monday, October 18th, 2021, 11:00pm in Virtual Campus
University of Ottawa - Université d'Ottawa

Jochen Lang

1 Texture Image Comparison

This assignment will explores template matching and the multilayer perceptron with texture data. The idea here is that we don't see all texture classes during training. Instead we want to find out if two texture images show the same texture or not. We will be using the Kylberg Texture Dataset v. 1.0 by Dr. Gustaf Kylberg [1] at the Centre for Image Analysis, Uppsala University, Sweden. The original database contains 28 texture classes and is available at `https://www.cb.uu.se/~gustaf/texture/`. However, for this assignment, we are only using the small subset of 6 classes with 40 images each. We will be using two subsets of images: 180 images for training and validation and 60 images for testing. Please note that these image sets are also available on BrightSpace as attachment to this assignment. You are not allowed to use the testset for anything else than for your final assessment of the approaches in Section 1.7.

1.1 Getting Started

You will need to download the two subsets from BrightSpace or from Uppsala University. Unpack the images in a directory relative to your jupyter notebook called `textures`. We will be marking your notebook with the data installed in `textures/training` and `textures/testing`, and your notebook will have to work with the images at these locations in the corresponding six subdirectories named for the texture (canvas1, cushion1, linsseeds1, sand1, seat2 and stone1). The training and validation data are the images numbered 001 to 030, while the testing images have the numbers 031 to 040. Do not rename images, directories or reorganize the data. You will loose marks if your notebook does not work with images at the expected locations.

1.2 Image Preprocessing [1.0]

You need to write a python function that loads images and preprocesses them as descibed below. Note that this function will have to mangle the filenames accordingly to find the desired pairs. The images are of size 576×576. Resize the images to a size of 32×32.

1.3 Image Matching [2.0]

Write a function `matchingImages` that accepts two of the images and calculates a score based on the similarity of the two images using either cross-correlation, convolution *or* sum of squared differences. Do not use any skimage function for this part. Your function should have the following list of parameters: `( imageA, imageB, method=cc|conv|ssd, normalize=y|n)` where the

optional argument method selects cross-correlation (cc), convolution (conv) or sum of squared differences (ssd). The method should default to non-normalized cross-correlation.

Given the similarity score, perform a simple comparison of two images using a threshold to decide if two images show the same texture. Build a simple classifier based on this score which returns true if two images match, i.e., the images pass the threshold. Evaluate your classifier with pairs chosen from the images in the training data set. There are a total of $\sum_{i=1}^{N-1} i = \frac{(N)(N-1)}{2} = \frac{(160)(159)}{2} = 12,720$ pairs in the training data set but most of the pairs will show different textures. Use a suitable fraction of the pairs for this validation step. Note your image matching approach will have no training step.

1.4 Perceptron [3]

Build a multilayer perceptron model (similar to the MNIST example shown in class) to classify an image pair as showing the same texture or different textures. Note that this is a binary classification. You will need to feed in pairs of images by concatenating them into your network. The simplest approach is to concatenate the pair of images ending up with two channels, i.e.,an image of size $32 \times 32 \times 2$ which then have to convert into a vector of size 2048. For this part of the assignment, you must build and train the Multi-layer perceptron model with scikit-learn, or alternatively with the Keras API of tensorflow. It may be useful to balance the dataset by sub-sampling the negative (i.e., different) class. Use the same validation set as in Section 1.3 to monitor the training of your classifier.

1.5 Classification Comparison [1]

Compare the classifier of Sections 1.3 and 1.4 on the test data subset. Consider classifier performance but also other criteria, e.g., training effort, prediction speed, generalization and robustness. Your brief discussion based on quantifiable criteria need to be contained in your Jupyter notebook.

1.6 Feature Engineering [1.5]

Considering the results for Sections 1.3 and 1.4 design an improved classifier that uses a multilayer perceptron for classification but first uses some form of feature extraction from an image pair. Hint: Have a look ar `skimage.feature` and `skimage.filter`. In general, you are allowed any skimage function for this part. For this part, you are not allowd more than 32 features as input to the MLP. Use the same validation set as in Section 1.3 to monitor the training of your classifier.

1.7 Discussion on Feature Engineering [1.5]

Briefly discuss why you expect your approach to improve the classification results and use the test data to show that your method is successful.

2 Submission

You will need to submit your solution in a Jupyter file, do *not* submit the image 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.

References

[1] G. Kylberg, *Kylberg texture dataset v. 1.0*. Centre for Image Analysis, Swedish University of Agricultural Sciences and Uppsala University, Sweden, 2011.