Menoufia University

Assignment 2

Template Matching && MLP

1 Texture Image Comparison

This assignment will explore 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 which is available at https://drive.google.com/drive/folders/1BxhLTMnzK4DyvULUA2cI8qXXlMDTF0iX?usp=sharing. We are using 6 classes with 40 images each. We will be using two subsets of images: 180 images for training and 60 images for testing.

1.1 Getting Started

You 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 data are the images numbered 001 to 030, while the testing images have the numbers 031 to 040.

1.2 Image Preprocessing

You need to write a python function that loads images and preprocesses them as described 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 [1.5 grade]

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

(180)(179)/2 = 16,110 pairs in the training data set but most of the pairs will show different textures. Use the test set to monitor the training of your classifier and print the **confusion matrix**. **Select the suitable evaluation metric from your choice to evaluate your performance (Accuracy, Recall, Precision, or F1 score),** you can use sklearn library directly without manual computation. Note your image matching approach will have no training step.

1.4 Perceptron [1.5 grade]

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*32*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 tensorflw. It may be useful to balance the dataset by sub-sampling the negative (i.e., different) class. Use the test set as in Section 1.3 to monitor the training of your classifier and print the **confusion matrix**. **Select the suitable evaluation metric from your choice to evaluate your performance (Accuracy, Recall, Precision, or F1 score),** you can use sklearn library directly without manual computation.

1.5 Classification Comparison [1 grade]

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 grade] + Bonus (2 grade)

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 at skimage.feature and skimage.filter. In general, you are allowed any skimage function for this part. For this part, you are not allowed more than **32 features** as input to the MLP. Use the same testing set to monitor the training of your classifier and print the **confusion matrix**. **Select the suitable evaluation metric from your choice to evaluate your performance (Accuracy, Recall, Precision, or F1 score),** you can use sklearn library directly without manual computation. **Briefly discuss** why you expect your approach to improve the classification results and use the test data to show that your method is successful.