Bank Note Recognition Project:

Authors:

Shaneer Luchman - 216003502

Daniel Mundell - 220108104

Siphokazi H. Nene - 216009670

Akaylan Perumal - 216035695

Download:

The pdf report: https://drive.google.com/file/d/1jQ-N5qTqyPfBwnr-n60_cOk_eC2DkBmZ/

The python source code: https://drive.google.com/file/d/14jMwmb1K-6GrcvVdGXohGAio9Xra2gg8/

Unzip:

Extract the "BankNote" folder from Zip folder

Open PyCharm

Click File | Open | And select the "BankNote" folder

How to run:

Open the main.py file.

At the top of the file, there are two variables “print_correct” and “print_incorrect” which determine what information is printed when the file is run. Since it can be difficult to look for the incorrect bank notes in a list of 118, setting “print_correct” = False and “print_incorrect” = True, will make it easier to see what went wrong with the incorrect predictions.

Run the main.py file to classify all the bank notes present in the “RandNotes” folder.

Information on each bank note (according to the print_correct and print_incorrect variables above) will be printed out, showing what the actual note information was and what the system classified it as, along with a comparison between these two to show if the system was correct or incorrect.

The system will also print out a total number of bank notes scanned, along with the number of correct and incorrect classifications.

Here is an example of the output for an incorrectly classified note.

Image 41: 050BNR2.jpg				
	Actual	Predicted	Result	
Genuine:	True	False	**False**	
Denomination:	050	-	**False**	
Colour:	050	050	True	
GLCM:	050	200	False	0.591479
Front/Back:	Back	Back	True	
Old/New:	New	New	True	
Rotate:	Rotate	Rotate	True	

The two bold cells in the Result column determine whether the system classified the bank note correctly as genuine/non-genuine and as the correct denomination respectively. In this case, it can be seen that while the system correctly classified the Colour, Front/Back, Old/New, and Rotation, it did not correctly classify the denomination based on the Grey Level Co-Occurrence Matrix (GLCM), and therefore classified the note as non-genuine when in fact it was genuine.

How to re-train the neural networks:

Run the training.py file.

This will re-train both neural networks according to the images in the “training” folder, and save the corresponding weights to the following 4 files: nn_new_weights1.csv, nn_new_weights2.csv, nn_old_weights1.csv, and nn_old_weights2.csv.

Now run the main.py file to classify bank notes according to the new neural network weights.

Other information:

All common processing takes place in the processing.py file.

The values.py file contains the boundaries for segmentation of regions, and the mean colours per BGR channel for the k-NN.