Credit Hours System
CMPN405-Pattern Recognition

Cairo University
Faculty of Engineering

CMPN405-Pattern Recognition Project Report

Submitted by:

- Sandra hany 1180563
- Andrew amir 1180086
- Mohamed Ahmed Zaki 1180445

Submitted to:

Eng. Ahmed Salem

Eng. Hussein Ehab

a) Project pipeline

The project pipeline is that first we preprocess our images such that we convert them from jpg then to gray scale and finally into binary with a black background.

Now our images are ready to extract from it the features to classify them, so we pass these images to the lpq function that has 255 features.

Our images now are ready to be classified, we then classify them according to these 255 features using the OneVsRestClassifier.

Then we test this classifier on our training set and the test set (that was split from the training data).

b) Preprocessing Module

First, we convert the images from jpg format to gray scale, then we apply blur on the images to remove the noise, and then convert the images into binary format. Finally apply inverse to the image that is changing the color of the background from white into black and changing the color of the texts from black into white.

c) Feature extraction/selection

We used the lpq function that returns 255 features, as it converts the image to double then apply convolution on the image as it computes the gradient of an image by it, where horizontal operators are real and vertical operators are imaginary. Now we need to concatenate the frequencies that we extracted; we do this by function dstack. As it stacks arrays in sequence depth wise (along third axis) that is applying concatenation along the third axis. For example, if we apply dstack on array 1,2,3 and array 4 5 6 we get a 2-D array

[1,4]

[2,5]

[3,6]

Then we perform quantization on the result after applying dstack. The last thing that we normalize the histogram.

d) Model selection

We selected the OneVsRestClassifier passing parameters C = 3000 and gamma = 50. This resulted in accuracy 99.89% on the training set and 99.85% on the test set that got split. Moreover, we used KNeighborsClassifier with n = 3

e) Performance analysis module

The OneVsRestClassifier resulted in accuracy 99.89% on the training set and 99.85% on the test set that got split.

The KNeighborsClassifier resulted in accuracy 86.35%

f) BONUS

Comparison between OneVsRestClassifier and KNeighborsClassifier

OneVsRestClassifier	KNeighborsClassifier
Transforms a multiclass problem into C binary classification problem as we will be building C different binary classifiers. As here we pick one class and train a binary classifier with samples of selected class on one side and other samples on the other side. Therefore, we end up with C classifiers. To test we classify the samples as belonging to the class with maximum score among the C classifiers. For example, if we have classes 1, 2, and 3. We copy our datasets 3 times. And for our first copy we replace labels 1 and 2 by 0. And for our second copy we replace classes 1 and 3 by 0. And for the last copy we replace labels 1 and 2 by 0. At this moment we have 3 binary classification problems as we have to distinguish between labels 1 and 0, 2 and 0, and 3 and 0, finally we apply the three classifiers to the input, and we get three predictions, and we return the prediction of a non-zero class which is the most certain. We settled the c with 3000, where the c parameter trades off correct classification of the training data against the maximization of the decision function's margin, as when the c is large, this results in a decision function that has small margin; on the other hand, when the c is small, the margin is large, and our decision function is more simple than the one that has a large value of c. The parameter gamma defines the level of influence of a single training example reaches, as when gamma is large this means that the level of influence of the point is low, and when the gamma is low this means that the level of influence is high.	The K nearest neighbors classifier is a straightforward classifier that is used to extend to multiclass cases. As we find the nearest k examples and return the class that was the most among the k examples. The parameter used here is K, where k is the number of examples that we will consider when classifying our test point. For example, when k = 3, this means that we are selecting the nearest 3 examples from our case and classify our point according to the class that repeated the most among the 3 examples.