UNIVERSITAT
ROVIRA i VIRGILI

Artificial Vision and Pattern Recognition

Lab Assignment 2

Student:

Giorgio Rossi

December 1, 2020

Contents

1 Introduction

I have collected all the material related to this laboratory, code, paper and report on a Github repo, available at `jeorjebot/action_recognition_urv`.

The algorithm is coded on a Matlab Live Script, same as the previous assignment, in order to visualize better the output and have a nice GUI.

2 Implementation of the Algorithm

2.1 Algorithm phases

I have implemented the four phases described in the assignment, plus some other phases useful to the functioning of my algorithm. All the phases are the following:

1. **Struct Creation phase**, that takes in input a proper directory of the dataset, such as `TrainSet` or `TestSet`, and creates a structure with useful informations for the other phases, such as for each image the class and the path. This phase is coded in the `create_struct` MatLab function and it is computed on the train set and the test set.

2. **Preprocessing phase**, in which the images are smoothed, resized (256x256) and the illumination is normalized with the Histogram Equalization (HE) [1] technique. I personally have skipped the smooth step since without smooth I have achieved better results. This phase is coded in the `preprocessing` and `preprocess_image` functions.

3. **Features Extraction phase**, on which the HOG [2] and LBP [3] features are extracted from each image.

   I wanted to achieve similar dimensions from the LBP and HOG features extracted from each images, and since with the default parameters the HOG features have a dimensionality of **approx. 35000**, so I have modified the parameters of the LBP extraction method to achieve a similar dimensionality. In fact I have set the `CellSize` parameter to 32, and increased `NumNeighbors` and `Radius`, to encode more details.

4. **SVM Training phase**, on which I have trained the SVM classifier. I have tried firstly a training only with LBP features, then only with HOG features, and then with the concatenation of LBP and HOG features. I have used the `fitcecoc` function provided by MatLab.

5. **SVM Prediction phase**, on which I have predicted the class of the test set using the SVM trained in the previous phase. Accordingly to what I have written before, I have experimented firstly with only LBP features, then only with HOG features, and then with the concatenation of LBP and HOG features. I have used the `predict` function provided by MatLab.

6. **Metrics Computation phase**, on which the algorithm compute some metrics from the Confusion Matrix, such as the number of **correct classified and misclassified**, **accuracy**, **precision**, **sensitivity**, **specificity** and **F1 score**, for each class and for the overall prediction (using the mean due to the homogeneous number of test images for each class). This phase is coded in the `print_results` and the `compute_misclassified` functions.

2.2 Input

The algorithm requires no input, but needs to be placed in the same directory of the `Dataset` directory. So for starting the algorithm we have to create a directory and put inside the `Dataset` folder (with the two sets `TrainSet` and `TestSet`) and the MatLab (Live) Script `Action_Recognition.mlx`.

2.3 Output

The algorithm creates a `Preprocessed` directory with the images obtained from the Preprocessing phase, and gives in output the figure of the Confusion Matrix, a table with the metrics for each class and the metrics for the overall prediction.

3 Results

3.1 Metrics

I have reported in this table the 6 tests that I have conducted on my algorithm. The metrics are **correct classified**, **misclassified**, **accuracy**, **precision**, **sensitivity**, **specificity** and **F1 score**.

	+	-	**Acc**	**Prec**	**Sens**	**Spec**	**F1**
LBP1	27.14	72.85	79.18	27.46	27.14	87.85	0.266
LBP2	35	65	81.42	36.78	35	89.16	0.347
HOG1	38.57	61.42	82.44	38.59	38.57	89.76	0.380
HOG2	36.42	63.57	81.83	36.43	36.42	89.40	0.361
LBP2 & HOG1	39.28	60.71	82.65	39.61	39.28	89.88	0.389
LBP2 & HOG2	41.42	58.57	83.26	42.65	41.42	90.23	0.416

3.2 LBP1 Test

In this test I have used the default parameters of the `extractLBPFeatures` function. For each image I have obtained an array of **59** features and I have used only this features to train and test the SVM.

3.3 LBP2 Test

In this test I have changed the default parameters of the `extractLBPFeatures` function, setting `CellSize` to 16, `NumNeighbors` to 12, `Radius` to 3. For each image I have obtained an array of **34560** features and I have used only this features to train and test the SVM.

3.4 HOG1 Test

In this test I have used the default parameters of the `extractHOGFeatures` function. For each image I have obtained an array of **34596** features and I have used only this features to train and test the SVM.

3.5 HOG2 Test

In this test I have changed the default parameters of the `extractHOGFeatures` function, setting `CellSize` to 12. For each image I have obtained an array of **14459** features and I have used only this features to train and test the SVM.

3.6 LBP2 & HOG1 Test

In this test I have **concatenated** the features obtained in the **LBP2 Test** and **HOG1 Test** for the training and testing of the SVM. I have combined the two features since the dimension of each array of feature was similar.

3.7 LBP2 & HOG2 Test

In this test I have **concatenated** the features obtained in the **LBP2 Test** and **HOG2 Test** for the training and testing of the SVM. There are more LBP features than HOG features (the ratio is 0.4) but the resulting metrics are better!

3.8 Final Considerations

Further changes to the parameters did not change the results, so the best results are achieved by the LBP2 & HOG2 Test.
I have calculated the metrics with the following formulas:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{3}$$

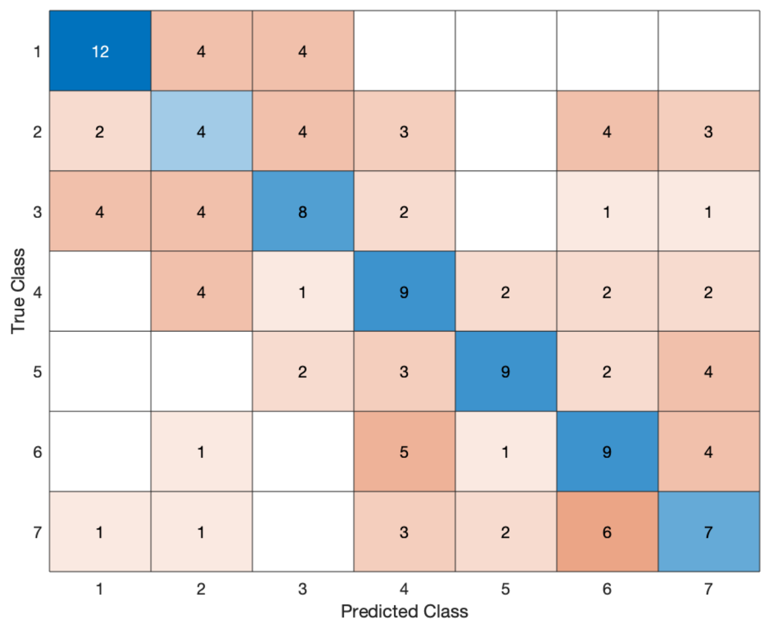

Figure 1: The Confusion Matrix of the final test

$$Sensitivity = \frac{TN}{TN + FP} \tag{4}$$

$$F1 - score = \frac{2TP}{2TP + FP + FN} \tag{5}$$

Where **TP** is **True Positive**, **TN** is **True Negative**, **FP** is **False Positive** and **FN** is **False Negative**.

References

[1] *Histogram Equalization*. URL: https://en.wikipedia.org/wiki/Histogram_equalization.

[2] *Histogram of Oriented Gradients - HOG*. URL: https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients.

[3] *Local Binary Patterns - LBP*. URL: https://en.wikipedia.org/wiki/Local_binary_patterns.