ROB311 - Apprentissage pour la robotique

TP4 - Facial Expression Recognition: KNN

Bastien Hubert and Michel Bitar
3rd Year - Robotics

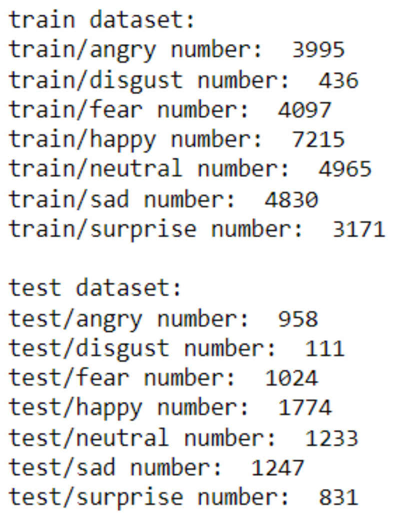
1 Introduction

In this document, we will classify the emotions given a static face image with KNN. The database contains 48×48 pixel grayscale images of faces divided into seven different categories:

$$\begin{aligned} 0 &= Angry \\ 1 &= Disgust \\ 2 &= Fear \\ 3 &= Happy \\ 4 &= Sad \\ 5 &= Surprise \\ 6 &= Neutral \end{aligned} \tag{1}$$

2 Datasets

Two datasets will be used in our experiment: train dataset and test dataset. The main difference between training data and testing data is that training data is the subset of original data that is used to train the model, whereas testing data is used to check the accuracy of the model. The training dataset is generally larger in size compared to the testing dataset as we can see in figure 1.



```
train dataset:
train/angry number:  3995
train/disgust number:  436
train/fear number:  4097
train/happy number:  7215
train/neutral number:  4965
train/sad number:  4830
train/surprise number:  3171

test dataset:
test/angry number:  958
test/disgust number:  111
test/fear number:  1024
test/happy number:  1774
test/neutral number:  1233
test/sad number:  1247
test/surprise number:  831
```

Figure 1: Training data and testing data

In order to check if we successfully imported the images, we use the package of matplotlib to show 5 images from each category:

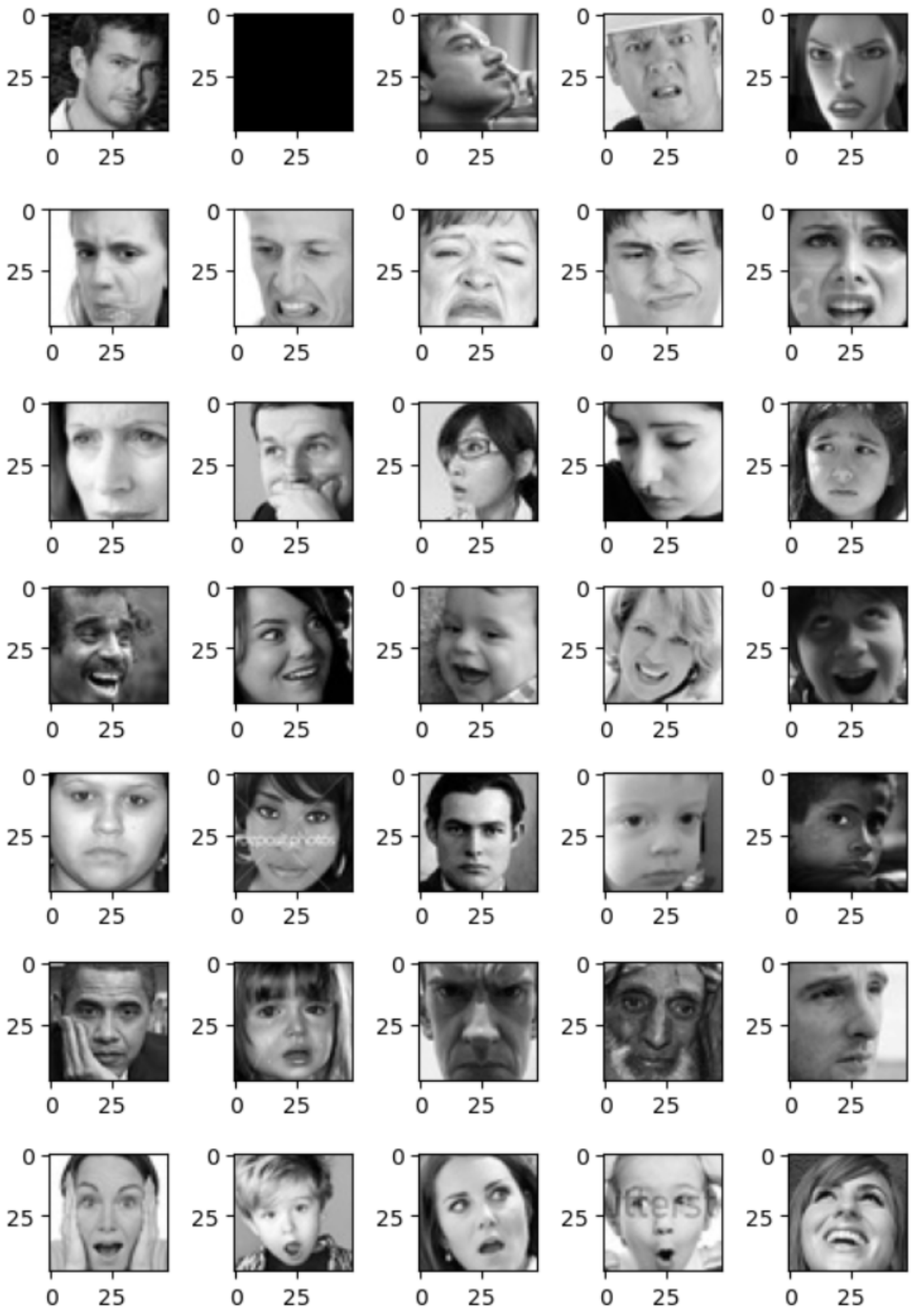


Figure 2: Train dataset

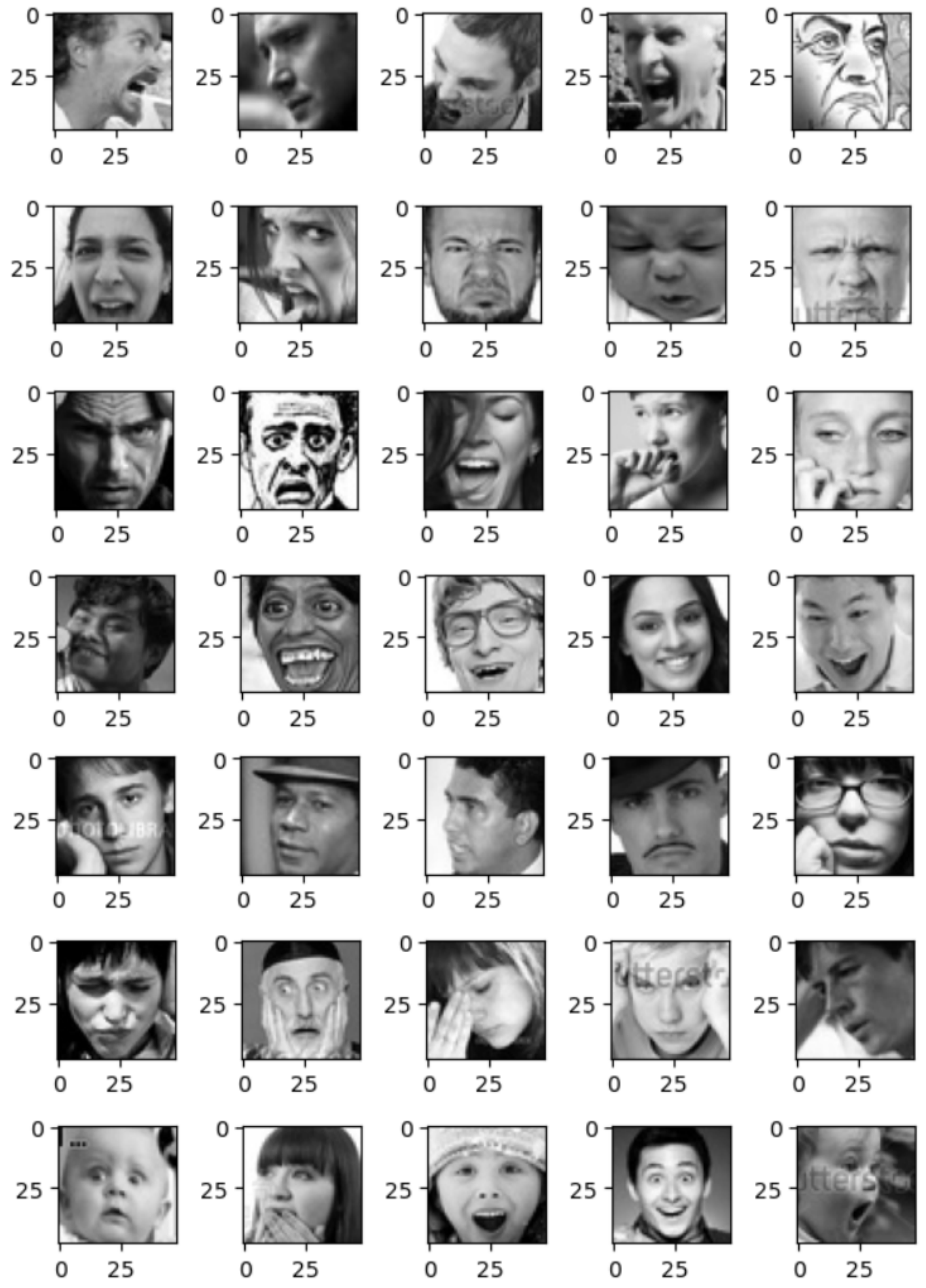


Figure 3: Test dataset

3 Local binary patterns (LBP)

Local binary patterns (LBP) is a type of visual descriptor used for classification in computer vision. The LBP feature vector, in its simplest form, is created in the following manner:

- Divide the examined window into cells.

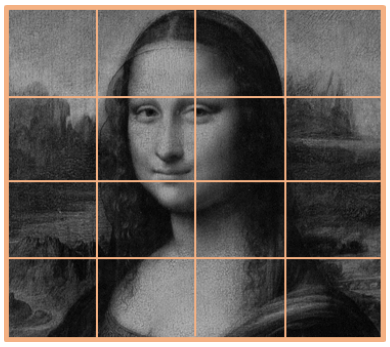


Figure 4: Division of the image

- For each pixel in a cell, compare the pixel to each of its 8 neighbors. Where the center pixel's value is greater than the neighbor's value, write ”0”. Otherwise, write ”1”. This gives an 8-digit binary number (which is usually converted to decimal for convenience).

w0.61w

Figure 5: Pixels value in each cell

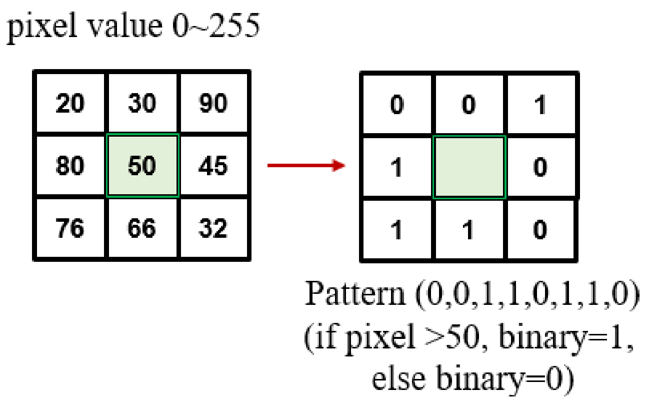


Figure 6: Replacing by binary numbers

- Compute the histogram, over the cell, of the frequency of each ”number” occurring. This histogram can be seen as a 256-dimensional feature vector.
- Concatenate histograms of all cells. This gives a feature vector for the entire window.

After applying the LBP method to the grayscale images in our database, the local texture features of the images will be extracted. We can show an example in figure 7.

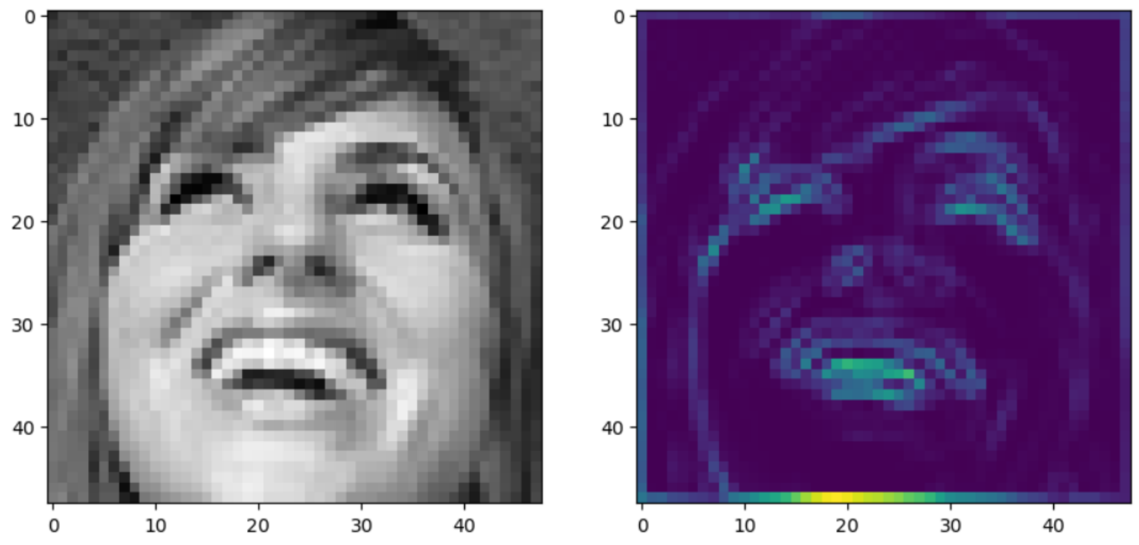
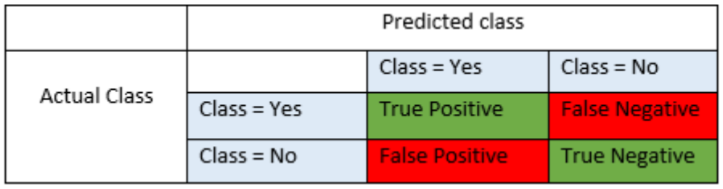


Figure 7: Application of LBP

4 Facial Expression Recognition: KNN

The K-nearest neighbors algorithm (KNN) is a non-parametric supervised learning method that is used for classification and regression. In both cases, the input consists of the K closest training examples in a data set. In KNN classification, an object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its K nearest neighbors. In order to evaluate the performance of our experiment in the datasets, we will use the classification_report from *sklearn.metrics* that will show us the accuracy, precision, recall and f1-score. Before that, we will explain the meaning of each parameter. Normally, we have a confusion matrix that includes TF (True Positive), FN (False Negative), FP (False Positive) and TN (True Negative). It is shown in figure 8.



		Predicted class	
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

Figure 8: Confusion matrix

- True Positives (TP): These are the correctly predicted positive values. E.g. if actual class value indicates that this image is in the happy category and predicted class tells you the same thing.
- True Negatives (TN): These are the correctly predicted negative values. E.g. if actual class

says this image is not in the happy category and predicted class tells you the same thing.

- False positives and false negatives, these values occur when your actual class contradicts with the predicted class.

Now that we know the parameters of the confusion matrix, we can explain the meaning of the classification_report's parameters:

- Accuracy: it is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \tag{2}$$

  Accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same. Therefore, you have to look at other parameters to evaluate the performance of your model.

- Precision: it is the ratio of correctly predicted positive observations to the total predicted positive observations.

$$Precision = \frac{TP}{TP+FP} \tag{3}$$

- Recall: it is the ratio of correctly predicted positive observations to the all observations in actual class.

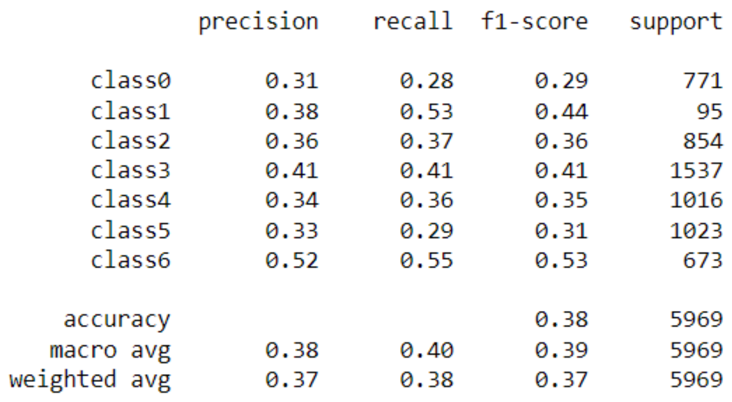
$$Recall = \frac{TP}{TP+FN} \tag{4}$$

- F1-score: it is the weighted average of Precision and Recall.

$$F1score = \frac{2 \times (Recall \times Precision)}{Recall + Precision} \tag{5}$$

  Therefore, this score takes both false positives and false negatives into account. It is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

The classification reports are shown below for different number of neighbors for KNN:

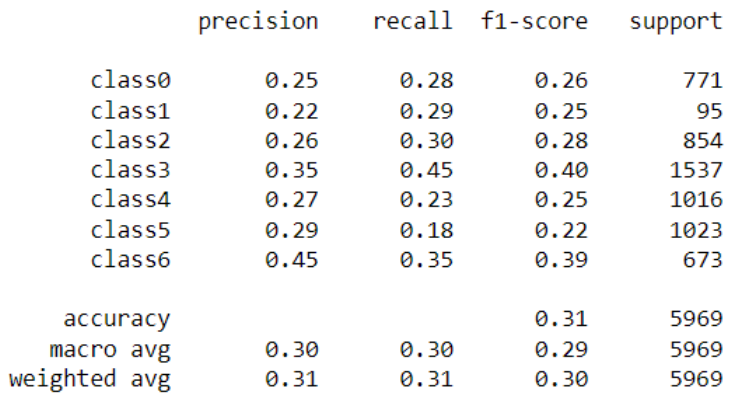


```
              precision    recall  f1-score   support

      class0       0.31      0.28      0.29       771
      class1       0.38      0.53      0.44        95
      class2       0.36      0.37      0.36       854
      class3       0.41      0.41      0.41      1537
      class4       0.34      0.36      0.35      1016
      class5       0.33      0.29      0.31      1023
      class6       0.52      0.55      0.53       673

    accuracy                           0.38      5969
   macro avg       0.38      0.40      0.39      5969
weighted avg       0.37      0.38      0.37      5969
```

Figure 9: KNN classification report with $K = 1$

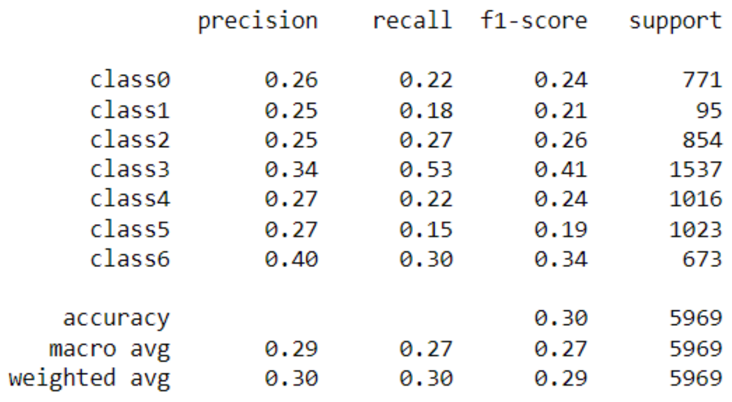


```
              precision    recall  f1-score   support

      class0       0.25      0.28      0.26       771
      class1       0.22      0.29      0.25        95
      class2       0.26      0.30      0.28       854
      class3       0.35      0.45      0.40      1537
      class4       0.27      0.23      0.25      1016
      class5       0.29      0.18      0.22      1023
      class6       0.45      0.35      0.39       673

    accuracy                           0.31      5969
   macro avg       0.30      0.30      0.29      5969
weighted avg       0.31      0.31      0.30      5969
```

Figure 10: KNN classification report with $K = 5$

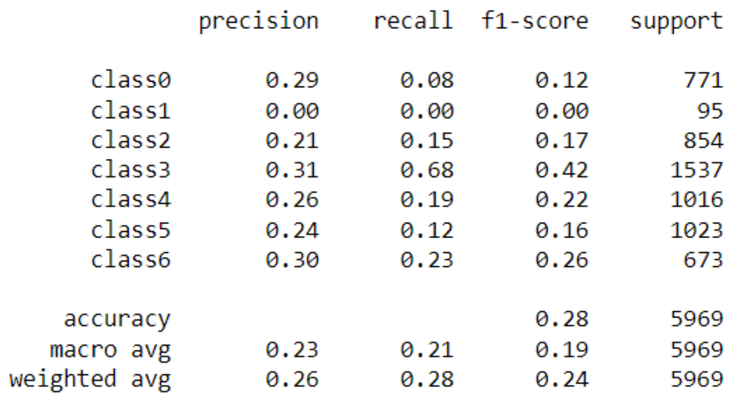


```
              precision    recall  f1-score   support

      class0       0.26      0.22      0.24       771
      class1       0.25      0.18      0.21        95
      class2       0.25      0.27      0.26       854
      class3       0.34      0.53      0.41      1537
      class4       0.27      0.22      0.24      1016
      class5       0.27      0.15      0.19      1023
      class6       0.40      0.30      0.34       673

    accuracy                           0.30      5969
   macro avg       0.29      0.27      0.27      5969
weighted avg       0.30      0.30      0.29      5969
```

Figure 11: KNN classification report with $K = 10$



```
              precision    recall  f1-score   support

      class0       0.29      0.08      0.12       771
      class1       0.00      0.00      0.00        95
      class2       0.21      0.15      0.17       854
      class3       0.31      0.68      0.42      1537
      class4       0.26      0.19      0.22      1016
      class5       0.24      0.12      0.16      1023
      class6       0.30      0.23      0.26       673

    accuracy                           0.28      5969
   macro avg       0.23      0.21      0.19      5969
weighted avg       0.26      0.28      0.24      5969
```

Figure 12: KNN classification report with $K = 50$

5 Conclusion

In this experiment, we used the KNN algorithm to solve classification problems. To judge the performance of the KNN algorithm, it was evaluated on two datasets. The first dataset was for training and the second was for testing. The final results verify that the KNN algorithm could not produce good classification results; the performance parameters are low. In addition, different K were used, but no significant changes in the results. In conclusion, LBP and KNN are not very effective on this database.