FACIAL EMOTION RECOGNITION WITH LBP, TEMPLATE MATCHING AND SVM

Tran Dinh Tu
University of Science
APCS
tdtu@apcs.vn

2nd Le Duy Chuong
University of Science
APCS
ldchuong@apcs.vn

3rd Dinh Le Quoc Viet
University of Science
APCS
dlqviet@apcs.vn

INTRODUCTION

In this paper [1], there are 2 approaches which are the local binary patterns with template matching and local binary patterns with SVM. In this final report, we summarize 2 approaches as well as the results we got.

I. METHODS

A. Local Binary Pattern

First, we implement the local binary pattern with the operator $LBP^{u2}_{8.2}$ which means our LBP use the uniform pattern and has 58 uniform labels and 1 uniform label. In [1], they normalize the faces to the fix distance between the centers of two eyes with 55 pixels, however, when we try to crop the picture it cannot cover all of the face, therefore, we modify the distance to 105 pixels. From that, our input image will be 245 * 210 (Figure 1).

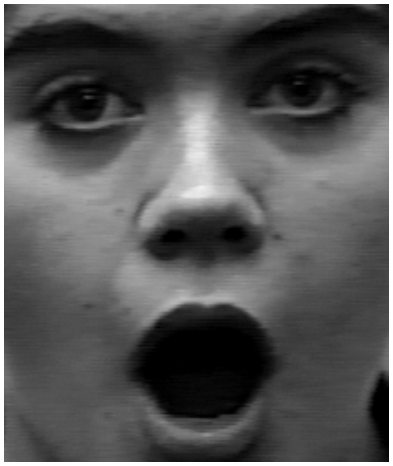

Fig. 1. Image example

In [1], they also divide the image into 42 regions (Figure 2)

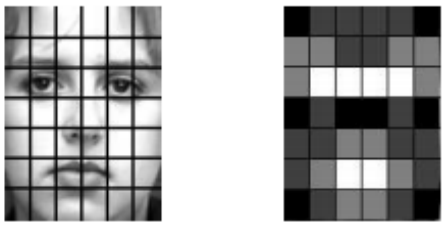

Fig. 2. Divided image example (left) and Weight example (right)

because they want to describe the micro-pattern of image especially regions of eyes and mouth containing a lot of information. Number 42 was chosen as they want the vector that represents image is not too long to get fast calculation as well as keep the high accuracy result. In our implementation, we have bigger image, therefore, the dimension of each region is 35x35. After getting the input, we divide the image into 59 regions. For each region, we calculate the LBP histogram of 59 bins. After that, we concatenated the vectors into a single vector, which has the length 59x32, representing our image. We have 6 labels of emotion of joy, disgust, sadness, fear, surprise and anger, each label is represented as a vector of length 59x32 which is the average of all LBP histogram of the training images set.

B. Template Matching

The template matching is used with weighted Chi-square to calculate the dissimilarity of the test image and the vectors that represent each label class. We have the equation:

$$X^2_w(S, M) = \sum_{ij} w_j \frac{(S_{ij} - M_{ij})^2}{S_{ij} + M_{ij}}$$

S, M are LBP histograms and W is the weight of each regions. Weight is calculated as Black squares indicate weight 0.0, dark gray 1.0, light gray 2.0 and white 4.0 (Figure 2 above). We set the weight like that because each region contains different

kind of information and just some of them are useful for our problem such as the region of mouth, eyes or eyebrows. For the input image, we calculate the LBP histogram then use the template matching with Chi-square to calculate the dissimilarity between the input image and the 6 arrays that represent 6 labels. After that, we choose the class which has the least dissimilarity compared to the input image (figure 3).

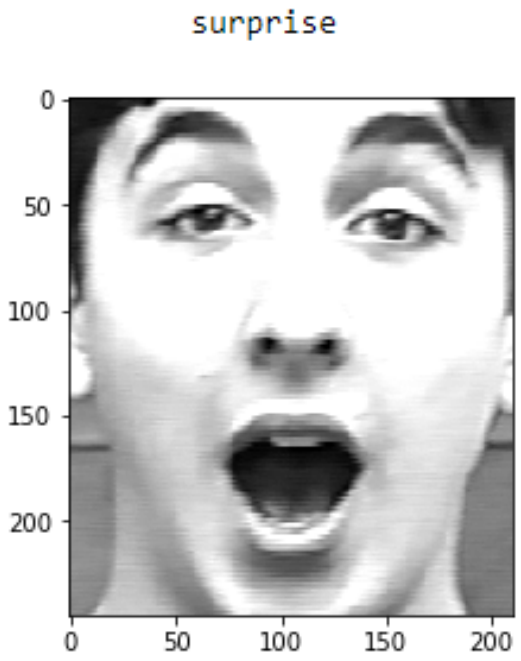

Fig. 3. The output of first approach example

C. Local Binary Pattern(LBP) and Support Vector Machine(SVM)

Instead of using template matching, we use SVM as a classifier. SVM performs an implicit mapping of data into a higher dimensional feature space, where linear algebra and geometry can be used to separate data which is only separable with nonlinear rules in the input space. Given a training set of labeled examples $T = \{f(x_i; y_i), i = 1, ..., l\}$ where $x_i \in R^n$ and $y_i \in \{1; -1\}$, the new test data x is classified by the following function:

$$f(x) = sgn(\sum_{i=1}^{l} \alpha_i \gamma_i K(x_i, x_j) + b)$$

where α_i are Lagrange multipliers of a dual optimization problem, and $K(x_i; x_j)$ is a kernel function. Given a nonlinear mapping ϕ that embeds input data into feature space, kernels have the form of $K(x_i; x_j) = (\phi(x_i) * \phi(x_j))$. SVM finds a linear separating hyperplane with the maximal margin to separate the training data in feature space. b is the parameter of the optimal hyperplane.

In SVM, we can choose the kernel for our method like linear, polynomial and RBF kernels. Because this is multi-classification, we use the one against rest method meaning we build k binary classifier of 1 label and the rest with k is the number of labels. After that, we get the one with the highest probability.

II. Dataset

Just like the paper [1], we use the same dataset of Cohn-Kanade Facial Expression Database [2]. However, with the protype, we use 420 images of 20 different people separated into 7 labels including of joy, disgust, sadness, fear, surprise, anger and neutral.

III. Implementation And Result

For the implementation, we use jupyter notebook environment with python 3.5, opencv 3.4.1, scikit-learn 0.20. We calculate the result in 2 cases, one is with the data that have neutral images and one without neutral images. We use the $Kfold = 10$ and calculate the average precision.

For the template matching we got the result (figure 4 and figure 5):

```
TRAIN: [1 2 3 4 5 6 7 8 9] TEST: [0]
PRECISION: 0.3611111111111111
TRAIN: [0 2 3 4 5 6 7 8 9] TEST: [1]
PRECISION: 0.4444444444444444
TRAIN: [0 1 3 4 5 6 7 8 9] TEST: [2]
PRECISION: 0.3055555555555556
TRAIN: [0 1 2 4 5 6 7 8 9] TEST: [3]
PRECISION: 0.5277777777777778
TRAIN: [0 1 2 3 5 6 7 8 9] TEST: [4]
PRECISION: 0.2222222222222222
TRAIN: [0 1 2 3 4 6 7 8 9] TEST: [5]
PRECISION: 0.3333333333333333
TRAIN: [0 1 2 3 4 5 7 8 9] TEST: [6]
PRECISION: 0.3888888888888889
TRAIN: [0 1 2 3 4 5 6 8 9] TEST: [7]
PRECISION: 0.2777777777777778
TRAIN: [0 1 2 3 4 5 6 7 9] TEST: [8]
PRECISION: 0.4444444444444444
TRAIN: [0 1 2 3 4 5 6 7 8] TEST: [9]
PRECISION: 0.3333333333333333

AVERAGE PRECISION: 0.3638888888888889
```

Fig. 4. 6-Class

```
TRAIN: [1 2 3 4 5 6 7 8 9] TEST: [0]
PRECISION: 0.3333333333333333
TRAIN: [0 2 3 4 5 6 7 8 9] TEST: [1]
PRECISION: 0.2619047619047619
TRAIN: [0 1 3 4 5 6 7 8 9] TEST: [2]
PRECISION: 0.21428571428571427
TRAIN: [0 1 2 4 5 6 7 8 9] TEST: [3]
PRECISION: 0.47619047619047616
TRAIN: [0 1 2 3 5 6 7 8 9] TEST: [4]
PRECISION: 0.19047619047619047
TRAIN: [0 1 2 3 4 6 7 8 9] TEST: [5]
PRECISION: 0.2857142857142857
TRAIN: [0 1 2 3 4 5 7 8 9] TEST: [6]
PRECISION: 0.35714285714285715
TRAIN: [0 1 2 3 4 5 6 8 9] TEST: [7]
PRECISION: 0.23809523809523808
TRAIN: [0 1 2 3 4 5 6 7 9] TEST: [8]
PRECISION: 0.38095238095238093
TRAIN: [0 1 2 3 4 5 6 7 8] TEST: [9]
PRECISION: 0.2857142857142857

AVERAGE PRECISION: 0.30238095238095236
```

Fig. 5. 7-Class

For the SVM we use the polynomial kernel with $degree = 1$ (figure 6 and figure 7):

```
TRAIN: [1 2 3 4 5 6 7 8 9] TEST: [0]
PRECISION: 0.8888888888888888
TRAIN: [0 2 3 4 5 6 7 8 9] TEST: [1]
PRECISION: 0.8888888888888888
TRAIN: [0 1 3 4 5 6 7 8 9] TEST: [2]
PRECISION: 1.0
TRAIN: [0 1 2 4 5 6 7 8 9] TEST: [3]
PRECISION: 0.9722222222222222
TRAIN: [0 1 2 3 5 6 7 8 9] TEST: [4]
PRECISION: 0.9722222222222222
TRAIN: [0 1 2 3 4 6 7 8 9] TEST: [5]
PRECISION: 1.0
TRAIN: [0 1 2 3 4 5 7 8 9] TEST: [6]
PRECISION: 0.9444444444444444
TRAIN: [0 1 2 3 4 5 6 8 9] TEST: [7]
PRECISION: 1.0
TRAIN: [0 1 2 3 4 5 6 7 9] TEST: [8]
PRECISION: 0.9722222222222222
TRAIN: [0 1 2 3 4 5 6 7 8] TEST: [9]
PRECISION: 0.9166666666666666

AVERAGE PRECISION: 0.9555555555555555
```

Fig. 6. 6-Class

```
TRAIN: [1 2 3 4 5 6 7 8 9] TEST: [0]
PRECISION: 0.8095238095238095
TRAIN: [0 2 3 4 5 6 7 8 9] TEST: [1]
PRECISION: 0.8095238095238095
TRAIN: [0 1 3 4 5 6 7 8 9] TEST: [2]
PRECISION: 0.9285714285714286
TRAIN: [0 1 2 4 5 6 7 8 9] TEST: [3]
PRECISION: 0.9285714285714286
TRAIN: [0 1 2 3 5 6 7 8 9] TEST: [4]
PRECISION: 0.9047619047619048
TRAIN: [0 1 2 3 4 6 7 8 9] TEST: [5]
PRECISION: 0.9285714285714286
TRAIN: [0 1 2 3 4 5 7 8 9] TEST: [6]
PRECISION: 0.9285714285714286
TRAIN: [0 1 2 3 4 5 6 8 9] TEST: [7]
PRECISION: 0.9285714285714286
TRAIN: [0 1 2 3 4 5 6 7 9] TEST: [8]
PRECISION: 0.9047619047619048
TRAIN: [0 1 2 3 4 5 6 7 8] TEST: [9]
PRECISION: 0.8333333333333334

AVERAGE PRECISION: 0.8904761904761906
```

Fig. 7. 7-Class

IV. Conclusion

The result of the SVM is good while the result of template matching is bad. Because of the fixed size of weighted Chi-square and the face in each image is not fixed in the suitable size, the result is not good. For getting better results in the future, we will try to auto detect the salient regions of a face.

References

[1] Robus facial expression recognition using local binary patterns Caifeng Shan, Shaogang Gong and Peter W. McOwan

[2] T. Kanade, J.F. Cohn, and Y. Tian, “Comprehensive databasefor facial expression analysis,” in IEEE FG, 2000, pp. 46–53