VNUHCM - University of Science
Faculty of Information Technology
Advanced Program in Computer Science

CS412 - Final Report
Face expression recognition

January 10, 2017

Group members:

1. 1351040 : Thai Thien
2. 1351059 : Ho Ngoc Huynh Mai

1 Introduction

We choose to do Face expression recognition. Our implement will based on paper Robust Facial Expression Classification Using Shape and Appearance Features of SL Happy and Aurobinda Routray [1]. We adopt the preprocessing process, the feature extraction. But we do not intend to work on extract active patches. For the model, we plan to implement 2 model, a Support Vector Machine (as in [1]) and a simple Neural network.

2 Project Details

The purpose of this project is implement the system to recognize facial expression, including, but not limited to anger, disgust, fear, happiness, sadness and surprise.
Input: A face of someone.
Output: The facial expression.

3 Methodology

3.1 Preprocessing

3.1.1 Gaussian blur

By default, Opencv load image with BGR channel. We convert image to grayscale. Then apply Gaussian Blur with kernel size 5x5 and $\sigma = 1$ to filter out noise.

3.1.2 Face Detection

The purpose of this step is to detect the face in input image. We use image which is smoothed from the step above as input for face detection. We use Haar feature-based cascade classifiers [2] for face detection.

We use pre-trained data provided by OpenCV [3] haarcascade_frontalface_default.xml.

3.1.3 Extract and resize

We create the region of interest (ROI) from the face position detected by Haar classifiers. We create new image from region of interest and resize the image to 96x96.

3.1.4 Result

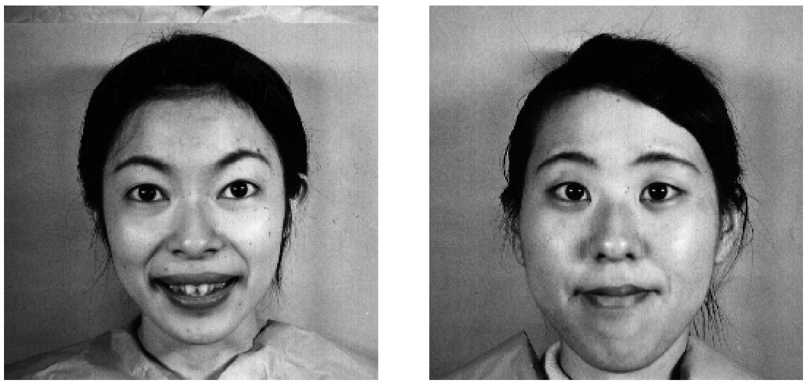


Figure 1: Before preprocessing

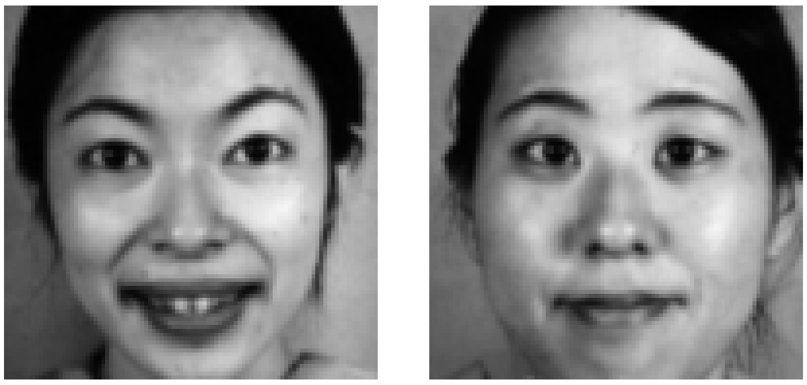


Figure 2: After preprocessing

3.1.5 Detect and extract facial patches

We use Haar feature-based cascade classifiers [2] with Opencv [3] pre-trained classifiers for eyes, nose and mouth. For each face, we extract two eyes, one nose, one mouth and save to same folder. For face we cannot extract 4 patches, we discard that sample.

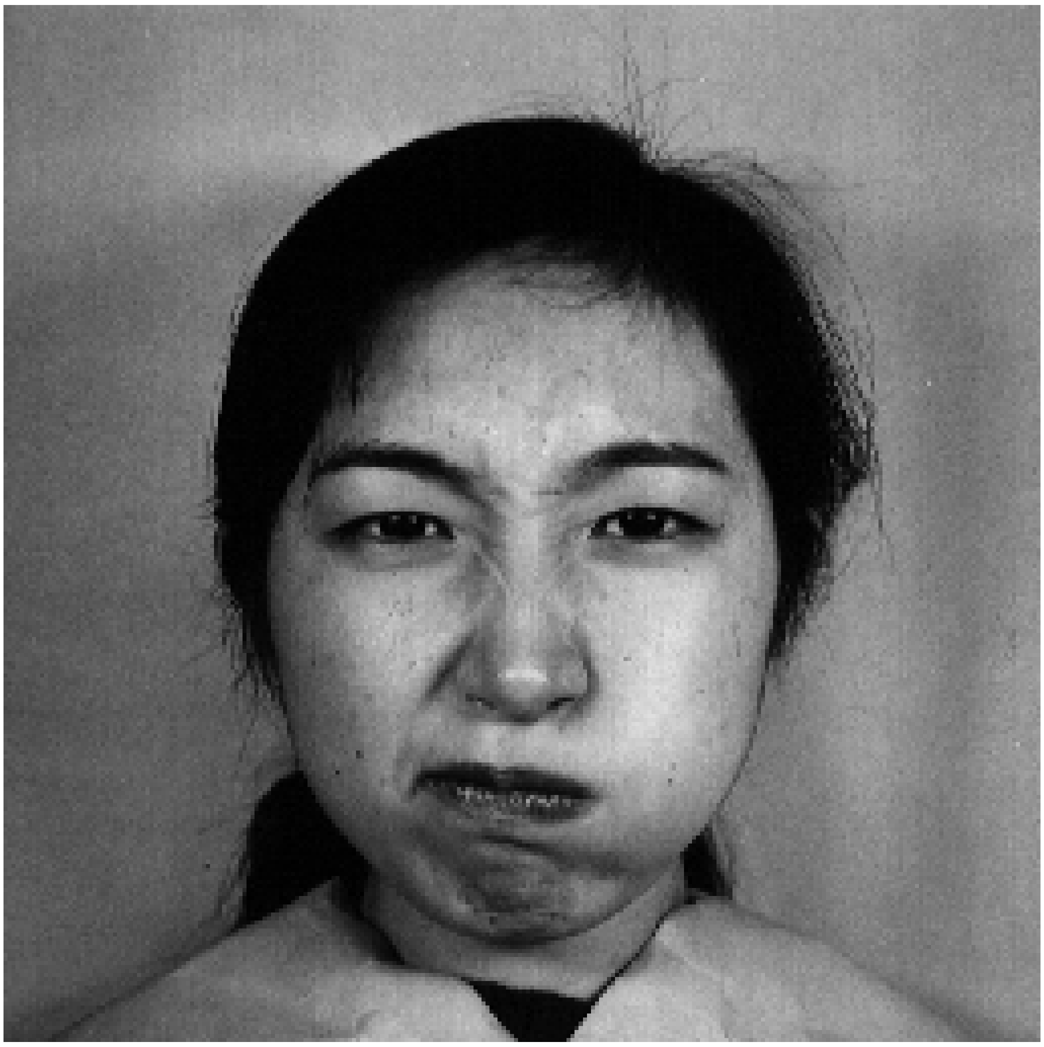


Figure 3: original image
[5]

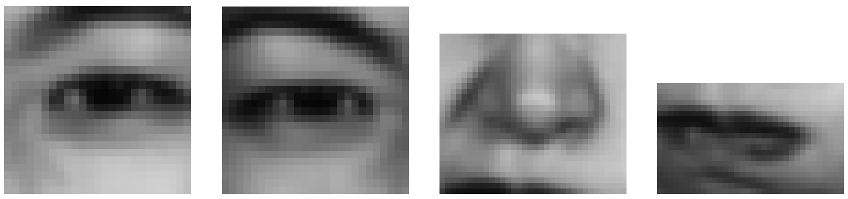


Figure 4: Facial Patches

3.2 Feature selection

This step we select feature to represent image.

Local Binary Pattern (lbp) is a type of texture descriptor[4].

Before construct lbp, the image must be convert into gray-scale. We already did that in 3.1

For each pixel in our image, we consider 8 neighbor pixels.

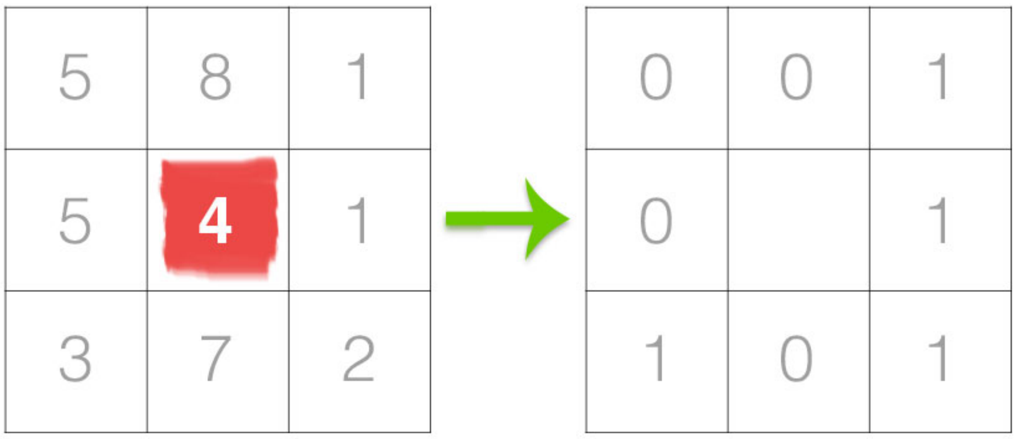


Figure 5: Local binary patterns (LBP)
[5]

We compare the intensitiy of the center pixel with its neighbor. Whenever the intensity of a neighbor pixel is greater or equal the middle pixel, we set to 1, else we set to 0. Then we take 8 bit represent of neighbor pixels and convert it into decimal representation. For example, in **figure 5**, we have $0010111_2 = 23_10$. Then we can store the values in another 2D array as in **figure 6**. There are 8 neighbors pixels, therefore, there are $2^8 = 256$ different pattern of lbp.

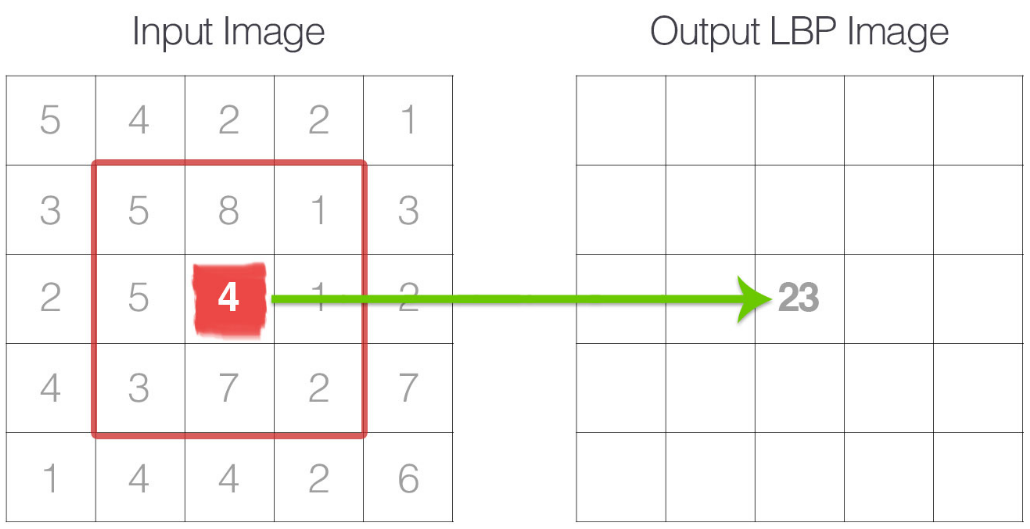


Figure 6: Local binary patterns (LBP)
[5]

Finally, we calculate the histogram of each pattern occur. We treat this histogram as our feature vector.

For Local binary patterns histogram (lbph) implementation, we use scikit-image library [6]

3.2.1 Feature extraction for full face

We take a full face after preprocessed in 3.1.1, 3.1.2, 3.1.3. Then we extract lbph feature from that image to make feature vector of a sample.

3.2.2 Feature extraction for facial patches

For each sample image, apply preprocess in 3.1.1, 3.1.2, 3.1.3 and 3.1.5 to get 4 patches. Then we extract lbph feature for each patches. We concatenate feature vectors of each patches to create feature vector of a sample.

3.3 The model

3.3.1 Support Vector Machine

The model is a classifier model, which take feature vector as a input and output the class it belong to. There are 6 classes anger, disgust, fear, happiness, sadness, surprise and neutral.

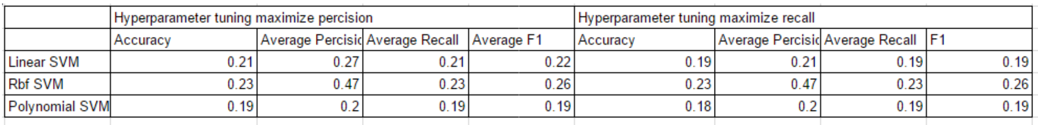
Then, we implement the One-Against-One Support Vector Machine. We need total 21 OAO SVM for 7 classes.

We use scikit-learn to [7] implement our SVM model. We use 3 different kernel: linear, polynomial, rbf. For each kernel, we run hyperparameter tunning for recall and precision.

3.4 Experiment result

3.4.1 Experiment 1

We set up the experiment for 3.1.1, 3.1.2, 3.1.3, 3.2.1, 3.3.1

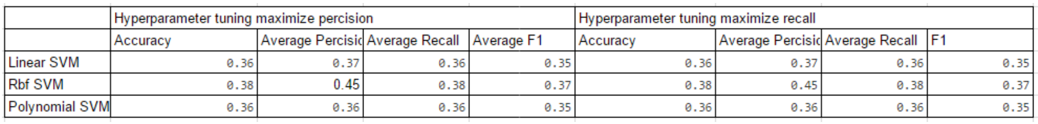


	Hyperparameter tuning maximize percision				Hyperparameter tuning maximize recall			
	Accuracy	Average Percisi	Average Recall	Average F1	Accuracy	Average Percisi	Average Recall	F1
Linear SVM	0.21	0.27	0.21	0.22	0.19	0.21	0.19	0.19
Rbf SVM	0.23	0.47	0.23	0.26	0.23	0.47	0.23	0.26
Polynomial SVM	0.19	0.2	0.19	0.19	0.18	0.2	0.19	0.19

Figure 7: Experiment result full face

3.4.2 Experiment 2

We set up the experiment for 3.1.1, 3.1.2, 3.1.3, 3.1.5, 3.2.2, 3.3.1



	Hyperparameter tuning maximize percision				Hyperparameter tuning maximize recall			
	Accuracy	Average Percisi	Average Recall	Average F1	Accuracy	Average Percisi	Average Recall	F1
Linear SVM	0.36	0.37	0.36	0.35	0.36	0.37	0.36	0.35
Rbf SVM	0.38	0.45	0.38	0.37	0.38	0.45	0.38	0.37
Polynomial SVM	0.36	0.36	0.36	0.35	0.36	0.36	0.36	0.35

Figure 8: Experiment result facial patches

4 Dataset

Dataset: jaffe [8]

References

[1] S. L. Happy and A. Routray, "Robust facial expression classification using shape and appearance features," in *Advances in Pattern Recognition (ICAPR), 2015 Eighth International Conference on*, pp. 1–5, Jan 2015.

[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–511, IEEE, 2001.

[3] G. Bradski *et al.*, “The opencv library,” *Doctor Dobbs Journal*, vol. 25, no. 11, pp. 120–126, 2000.

[4] M. Pietikäinen, “Local binary patterns,” *Scholarpedia*, vol. 5, no. 3, p. 9775, 2010.

[5] “Local binary patterns with python & opencv.” `http://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/`. Accessed: 2016-10-30.

[6] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, “scikit-image: image processing in python,” *PeerJ*, vol. 2, p. e453, 2014.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, “Scikit-learn: Machine learning in python,” *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[8] M. J. Lyons, S. Akamatsu, M. Kamachi, J. Gyoba, and J. Budynek, “The japanese female facial expression (jaffe) database,” 1998.