UROP 1100 Progress Report

Topic: machine learning on wearable devices

Supervisor: Prof. HUI Pan

December 22, 2017

LIU, Heshan

Project Abstract

The research aims to find a best algorithm to complete **the real-time emotion recognition** task **on the wearable devices** such as mobile phone and google glasses. In this research, we first design several machine learning emotion recognition algorithms based on some existing facial recognition algorithm and test them on some famous emotion dataset such as CK+ and Yale dataset to find the best algorithm. Then we try to implement it in android system to see whether it also works well in real-time recognition.

Some datasets used in the project:

1. **Kaggle facial expression dataset**

   The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. Each face is categorized based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

   The dataset consists of more than 30000 examples with corresponding emotion labels.

   Examples of dataset:

 Label: 5 (suprise)

 Label: 2 (fear)

1. **Cohn-Kanade database (CK and CK+)**

   The dataset consists of 327 sequences of images. Each sequence contains images from neutral face to the peak expression. The corresponding expression label is 0=neutral, 1=anger, 2=contempt, 3=disgust, 4=fear, 5=happy, 6=sadness, 7=surprise).

   Examples:

 to 

with label 1 (anger)

Comparison of Algorithms

Since the aim of the research is emotion recognition. We need first extract faces from the images and extract features from faces. Therefore, OpenCV library is used to extract faces and Dlib is used to extract features from faces.

We can also use the face recognizer and classifier integrated in OpenCV library to help us extract features and classify the emotions.

1. **Emotion classification using haarcascade classifier and FisherFaceRecognizer in OpenCV**

   Using OpenCV built-in recognizer fisher face recognizer to extract facial features and use haarcascade as classifier.

   (Using 5-fold-cross-validation)

Training set	Testing set	Testing accuracy
CK+ dataset	CK+ dataset	83.7%
kaggle dataset	kaggle dataset	About 25%

   Pros:

   The algorithm is easy to be implemented since we used the function in OpenCV library and the algorithm could get high accuracy in the CK+ dataset

   Cons:

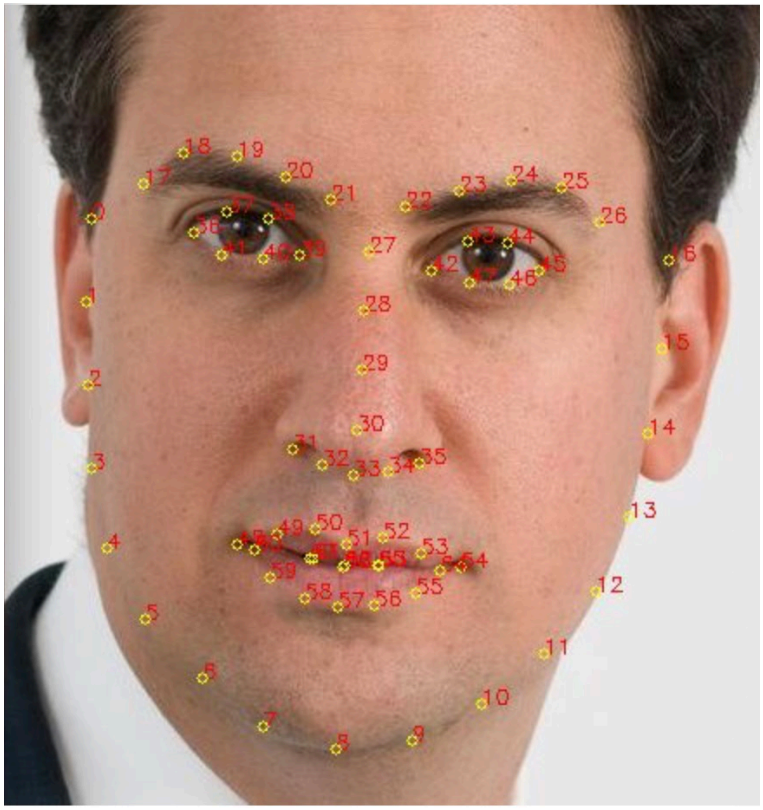
   The algorithm only works in relatively small dataset with clear image (such as CK+ dataset, with 640*490). If the image which captured by the camera is not clear enough, more

specifically, when the image's resolution is lower than 100*100 (in kaggle), it behaves really bad.

Brief summary:

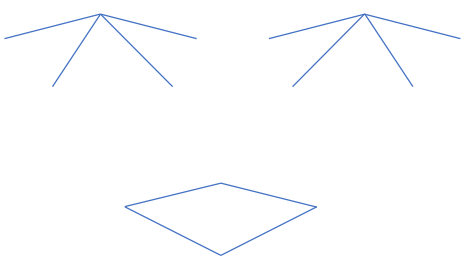
Although this algorithm works well on CK+ dataset, we need to figure a more stable one works in more datasets. Therefore, functions in Dlib become a good choice

2. **Emotion classification using feature vectors constructed by Dlib**



i. First of all, Dlib could help us get those 68 facial landmarks from a face image.

ii. We construct 12 key vectors using these key points by $V_k = (x_k, y_k)$ where $x_k = x_{start} - x_{end}$ and $y_k = y_{start} - y_{end}$



VECTOR NO.	STARTING POINT	ENDING POINT
1	17	19
2	21	19
3	24	22
4	26	24
5	51	48
6	54	51
7	57	54
…	…	…

iii. Then we normalize the vectors and use these normalized vectors as input data, emotion labels as training labels.

iv. We train and test the data using linear SVM and two-layer-neural network separately and got the following table

(Using 5-fold-cross-validation)

Training set	Testing set	Classifier	Testing accuracy
CK+ dataset	CK+ dataset	Linear SVM	72.3%
kaggle dataset	kaggle dataset	Linear SVM	38.7%
kaggle dataset	kaggle dataset	Two-layer neural network	43.9%

and the confusion matrix of using two-layer neural network on kaggle dataset

	ANGRY	**DISGUST**	**FEAR**	**HAPPY**	**SAD**	**SURPRISE**	**NEUTRAL**
ANGRY	562	159	122	83	0	144	574
DISGUST	65	22	8	16	0	15	69

FEAR	342	48	169	110	0	250	777
HAPPY	219	20	692	1719	0	74	603
SAD	454	47	98	76	0	119	834
SURPRISE	125	24	140	52	0	642	505
NEUTRAL	546	50	61	81	0	193	1501

We could see from the previous confusion matrix that, it's hard for the classifier to distinguish from fear to angry, from disgust to angry and from neutral to other emotions. And we see that the classifier ignores the "sad" label. After the observations of the vectors and original images. It's due to three reasons:

i. If we use the vectors as input features, the differences among vectors of fear, vectors of angry, vectors of sad are small.

ii. All of the emotions may have great opportunity to be misclassified as neutral since during training, neutral emotion can vary a lot.

iii. Although two images have the same emotion label, their normalized feature vectors may be very different since different people may have different facial features.

Brief summary:

Thus, generally, we find that the if we use the feature vectors as input features, it behaves better than the first algorithm, however, it may still have great opportunity misclassifying several emotions since feature vectors may vary a lot from different people even they have same emotions.

Then we consider another algorithm that classify everyone's emotion based on his/her own features.

3. **Emotion classification using the displacement of 68 key points between neutral and peak emotion:**

i. we set everyone's neutral face as base image, and then compute the distance of the 68 key points from neutral face to peak emotion face. We would get 68 distance data for each emotion.

ii. Normalized 68 distance data and use PCA to apply dimensionality reduction to the distance data.

iii. Using the distance data and corresponding emotion labels to train and test our model

to


with label 1 (anger)

(Using 5-fold cross validation)

Training set	Testing set	Classifier	Testing accuracy
CK+ dataset	CK+ dataset	Linear SVM	90.0%
CK+ dataset	CK+ dataset	Two-layer neural network	87.7%
Yale dataset	Yale dataset	Two-layer neural network	100%

and the confusion matrix for the CK+ dataset using Linear SVM classifier

1=anger, 2=contempt, 3=disgust, 4=fear, 5=happy, 6=sadness, 7=surprise

	ANGRY	**CONTEMPT**	**DISGUST**	**FEAR**	**HAPPY**	**SADNESS**	**SURPRISE**
ANGRY	36	2	7	0	0	0	0

CONTEMPT	1	12	2	1	0	1	1
DISGUST	10	0	49	0	0	0	0
FEAR	1	0	0	17	3	2	2
HAPPY	0	0	1	1	67	0	0
SAD	1	2	0	0	0	25	0
SURPRISE	0	1	0	1	0	1	80

We could see from the confusion matrix that, using the distances of neutral face and peak emotion face from the same person as input feature behaves very well in CK+ dataset. Although that the disgust may still be misclassified into angry, it is acceptable.

Pros:

The algorithm works well in CK+ dataset and Yale dataset, and it is computationally efficient since we could use L1 distance of the points

Cons:

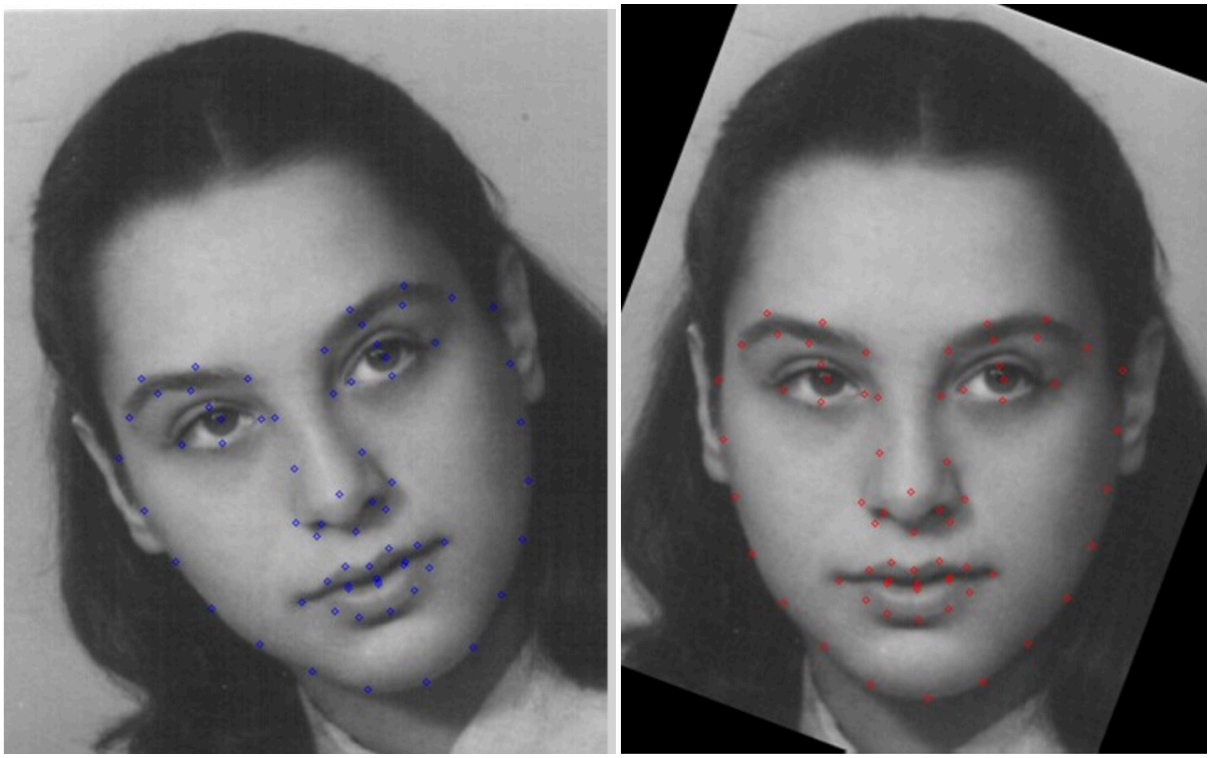
The algorithm can not be tested in kaggle data set since this algorithm need one's neutral face as base image. Also, the implementation of the algorithm would be more complicated since we need to capture one's neutral face first.

Implementation of the chosen algorithm

After comparing the testing accuracy in different dataset from the above algorithms, the 3rd algorithm works best and would be applied to our software. We used Dr. Paul Ekman's six basic emotions (happiness, sadness, anger, fear, surprise and disgust) plus one extra categorization neutral for people who are not revealing any hint of thoughts or feelings.

1. **Extracting face and affine transformation**

i. First of all, we use OpenCV to extract face from captured images and use Dlib to detect 68 key points from the face image.



ii. Apply affine transformation to the face image based on the 31st and 36th key points:

$$\begin{matrix} x' \\ y' \\ 1 \end{matrix} = \begin{matrix} m00 & m01 & m02 \\ m10 & m11 & m12 \\ 0 & 0 & 1 \end{matrix} * \begin{matrix} x \\ y \\ 1 \end{matrix}$$

where m00 = cosθ, m01 = sinθ, m02 = (1-cosθ)*mid.x - sinθ*mid.y

m10 = -sinθ, m11 = cosθ, m12 = sinθ*mid.x + (1-cosθ)*mid.y

dy = landmarks[36].y - landmarks[31].y

dx = landmarks[36].x – landmarks[31].x

θ = arctan(dy,dx) * 180 / π

mid.x = (landmarks[36].x + landmarks[31].x)/2

mid.y = (landmarks[36].y + landmarks[31].y)/2

2. **Capturing the neutral face (Still in progress)**

   From common sense we know that one would not hold his facial emotion for a long time, which means that most of time the face image we captured from a person should be neutral. Therefore, based on this common sense, we may assume that if the difference of the captured face in a continuous time period is smaller than a threshold T (Still need more test to figure out a proper T), we can set this captured face as the neutral face and apply the chosen algorithm

Summary

This report has presented several different emotion recognition algorithms that could be applied on wearable devices. We have compared their testing result in different dataset and their pros and cons to choose a best one. The machine learning algorithm introduced above require relatively small amount of computing power and the best one works pretty well in the real-time emotion recognition (above 85% accuracy). With the increasing computing power of mobile devices, we could try more deep learning algorithm for the emotion recognition on mobile devices, which could achieve higher accuracy than simple machine learning algorithms theoretically.