WROCLAW UNIVERSITY OF SCIENCE AND TECHNOLOGY

FACULTY OF INFORMATICS AND MANAGEMENT

FIELD OF STUDY: COMPUTER ENGINEERING

ADVANCED TOPICS IN ARTIFICIAL INTELLIGENCE

Automatic Face Expression Recognition

AUTHOR:
Mateusz Świerczyński

WROCŁAW 2018

Table of contents

1. Introduction

Facial expression is a visible representation of current emotional state. It expresses not only our emotions but also provides important clues for social interactions. According to psychologists, almost 60% of the communicated message is transmitted by facial expression. Voice and language constitute remained 40%. Nowadays scientists try to improve the interaction between humans and computers and, because of the reasons mentioned above, they probably will not be successful with simple voice recognition. Automatic face expression recognition is a key feature during communication between humans and artificial intelligence.
It can be also useful in many other applications, for example in anthropology. Emotion estimated by a computer is usually more objective than those performed by a human and can be used in clinical psychology, psychiatry and neurology. Customer's facial expression can also be collected by service providers as an implicit user's feedback to improve the quality of their services. A block diagram presenting steps from fetching a picture by a computer software to predicting a face expression can be seen below.

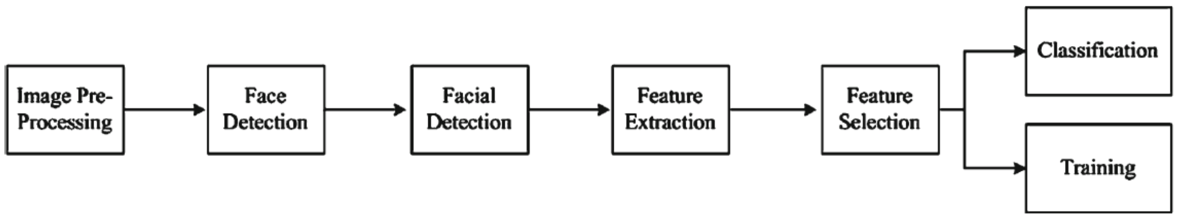


Figure 1. Block diagram of the facial expression recognition system. [https://link.springer.com/article/10.1007/s11760-010-0177-5]

2. The Dataset

To teach the artificial intelligence predicting readable by humans facial expression labels we need a dataset. In this case ‘teaching’ means providing to an algorithm series of pictures presenting different face expressions with corresponding labels, such as ‘happiness’ or ‘anger’. After detecting a face region from the whole picture, main features can be extracted from it. These features involve these parts of our face which allow humans to distinguish other people’s face expressions. As an example we can imagine how different will be the height and width of our mouth when try to smile and when we want to express anger. The better the dataset will be prepared, the easier and more accurate the face expression prediction will be.
In the Internet there are several sites with photo sessions of different people to which labels with corresponding emotions are attached but most of them need a written permission of their authors to download them or even we need to pay for them. One of the best dataset which can be found comes from University of Pittsburgh. It is named *The Extended Cohn – Kanade Dataset(CK+)*, after its co – founders: Jeffrey F. Cohn and Takeo Kanade. They work as University Professors at the University of Pittsburgh.

This dataset is free of charge and includes 593 sequences. Each of the sequence contains images from ‘neutral’ frame to peak expression. These photos present facial behavior of 120 adults between 18 and 50 years of age, 69% female, 81% Euro – American, 13% Afro – American and 6% other groups. These pictures comes with high resolution so for project purposes they were resized to 640x480.

Each person was pleased to present 8 different emotions: *Contempt, Disgust, Fear, Anger, Happy, Sadness, Surprise and Neutral.* Unfortunately not every sequence was successful. Ready – to – use dataset consists of 327 sequences. Rejected sessions were ambiguous and could affect significantly decrease of future facial expression prediction. Sample pictures of the dataset can be seen below.

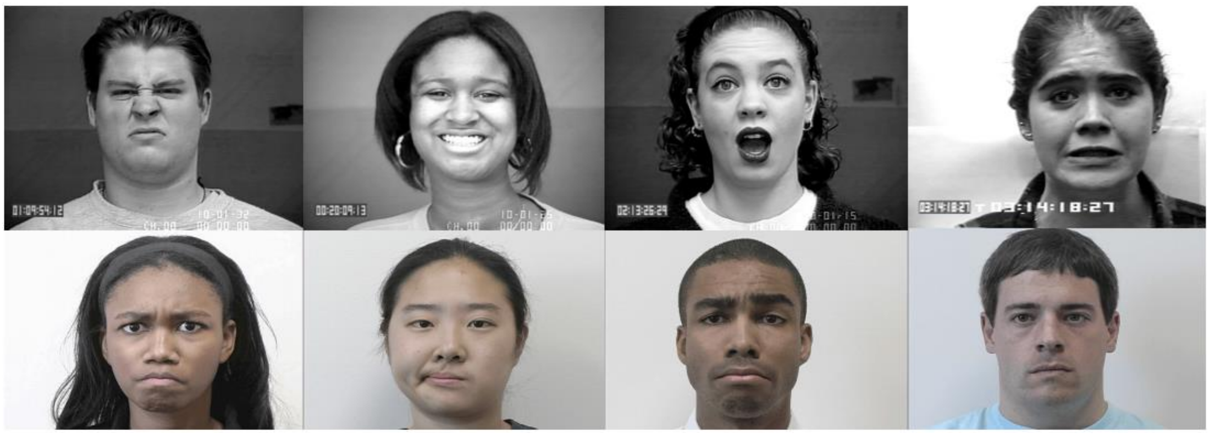


Figure 2. Examples of CK+ database. Presented emotions: a) disgust, b) happy, c) surprise, d) fear, e) angry, f) contempt, g)sadness, h) neutral. [http://www.consortium.ri.cmu.edu/data/ck/CK+/CVPR2010_CK.pdf]

Finally, after deleting manually some sequences because of their similarity to other emotions and taking 5 last pictures from remained series, a training set of 1200 photos has been prepared. This set will be then used for training the artificial intelligence.

3. Image preprocessing

The next step after the dataset preparation is the image preprocessing. In each photo we can see not only face, but the whole head with a small part of trunk and a background. For face expression these features are irrelevant and need to be cut off. This can be done by in – built tools from the open source library called OpenCV. It is dedicated for static image processing as well as real time processing. OpenCV comes preinstalled with a range of sophisticated classifiers for general – purpose object detection. The most commonly known detector is the cascade of Haar – based feature detectors for face detection. This classifier sums up the pixel intensities in small regions of an image, as well as captures a difference between adjacent image regions. For this project a pre – trained classifier for frontal face detection has been used. It is called *haarcascade_frontalface_default.xml*. Before face detection the photos are transformed to grayscale. If the face is detected, it is then cut from the origin photo, resized to *250x250* and appended to an output array of faces. Finally, all detected faces from the array with corresponding labels, representing emotions, are saved as a binary file for future processing. Two sample pictures after face detection and resizing can be seen below.

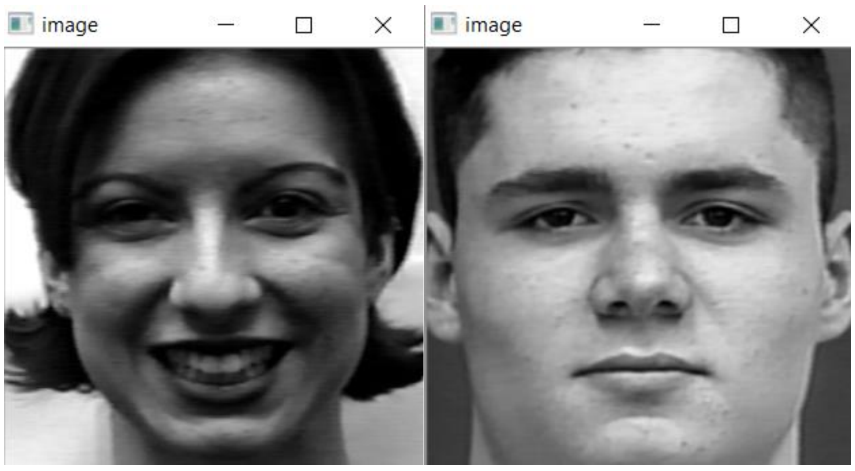


Figure 3. Sample picture after face detection and resizing

4. Feature extraction

An essential part of the entire learning task is finding the features that best describe the data. One of the best method is Principal Component Analysis (PCA).
PCA is a dimensionality reduction technique that is helpful whenever we are dealing with high – dimensional space. PCA extracts the most important features of a dataset. A simple example can be 2D set of points. It can be seen in the image below.

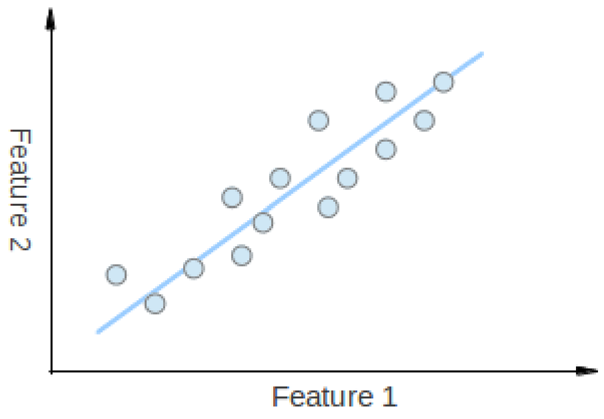


Figure 4. 2D set of points [https://docs.opencv.org/3.1.0/d1/dee/tutorial_introduction_to_pca.html]

Each dimension corresponds to one feature. If we look closely at the given points we can see that there is a linear pattern, expressed by a blue line. As it was mentioned, a key point of PCA is a dimensionality reduction. It is a process of reducing the number of dimensions of the given dataset. In the given example, the set of points can be well – mapped by a single line. It reduces the dimensionality of given dataset from 2D to 1D.
PCA allows to find the direction along which the data varies the most. The result of running PCA on the 2D set of points is a set of 2 vectors, called eigenvectors, which are the principal components of the dataset. These vectors begin in center of all points in the dataset. If we apply PCA to N – dimensional dataset we get N N – dimensional eigenvectors and one N – dimensional center point.
Eigenvectors produced as a result of PCA of given dataset can be seen below.

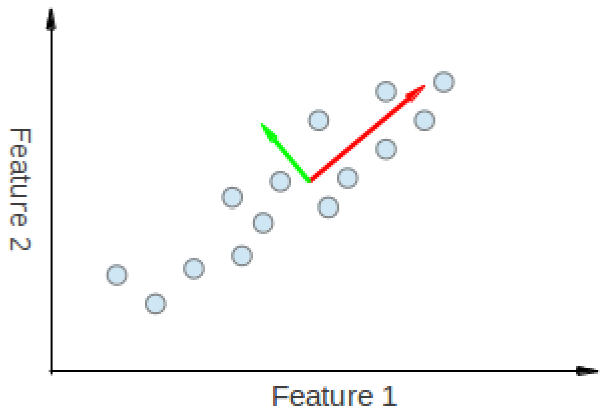


Figure 5. PCA results of 2D dataset

The process of computing eigenvectors and eigenvalues consists of few steps, described below.

4.1 Organizing the dataset

The goal is to transform a given dataset X of dimension p to an alternative dataset Y of smaller dimension L. To proceed, we need a set of n data vectors x_1 ... x_n in which each x_i represents a single grouped observation of p variables. In our case the photos with detected faces have been resized to 250 x 250 and flattened what means, that from 1200 pictures of size 250 x 250 we achieved 1200 single – dimensional vectors of length 250 x 250 = 62500. Each value from this range represents one pixel of an origin image and has a value between 0 to 255 (standard grayscale image color depth). After that we need to write these vectors as rows, one by one. Each vector has p columns (62500 in our case). After that these vectors can be simply transformed into a matrix of dimensions n x p.

4.2 Calculating the mean value

The next required step concerns computing a mean value along each dimension $j = 1 \ldots p$. Calculated values are then placed into a mean vector U of dimensions p x 1. Vector U can be described using the following formula:

$$U[\, j\,] = \frac{1}{n}\sum_{i=1}^{n} X[i, j] \tag{1}$$

4.3 Calculating the deviation of mean value

Next step will center the given data around the mean value. This is done by subtracting the mean vector U from each row of matrix X. The data after subtraction will be stored in new matrix B. It can be described with the formula:

$$B = X - hU^T \tag{2}$$

where h is an *n* x *1* vector of all 1s:

$$h[i] = 1, \quad i = 1, 2, \ldots, n \tag{3}$$

4.4 Finding a covariance matrix

In this step we will find a covariance matrix *C* from matrix *B*:

$$C = \frac{1}{n-1} B^* \cdot B \tag{4}$$

where * is a conjugate transpose operator (first transposes a matrix and then takes a complex conjugate of each entry). If a matrix *B* consists only of real numbers, not complex numbers, the conjugate transpose will be just a regular transpose. It is true in our case so finally the covariance matrix can be described as:

$$C = \frac{1}{n-1} B^T \cdot B \tag{5}$$

Matrix B^T will have dimensions *p* x *n*, matrix B *n* x *p* so the output matrix *C* will be a *p* x *p* matrix. It will look like a matrix below:

$$\begin{matrix} {\sigma_1}^2 & \sigma_{12} & \dots & \sigma_{1p} \\ \sigma_{21} & {\sigma_2}^2 & \dots & \sigma_{2p} \\ \vdots & \dots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \dots & {\sigma_p}^2 \end{matrix} \quad (6)$$

4.5 Finding eigenvectors and eigenvalues of covariance matrix

In the final step we compute the matrix *V* which diagonalizes the covariance matrix *C:*

$$V^{-1}CV = D \quad (7)$$

where D is the diagonal matrix of eigenvalues of C.

Matrix D will have dimensions p x p. The values in this matrix will be as follows:

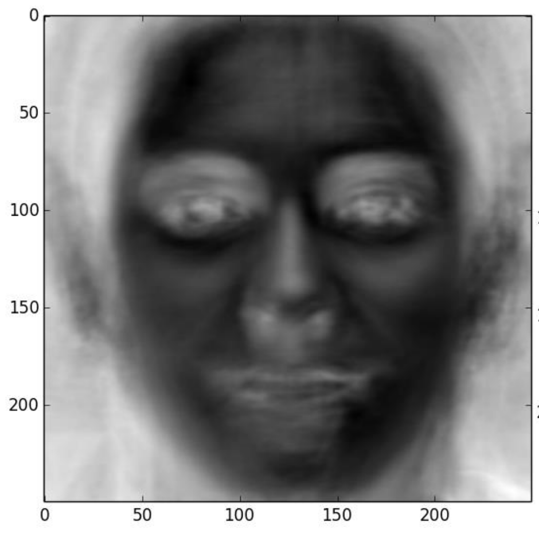
$$D[k,l] = \begin{cases} \lambda_k, k = l \ (diagonal\ of\ matrix) \\ \quad 0, \qquad k \neq l \end{cases} \quad (8)$$

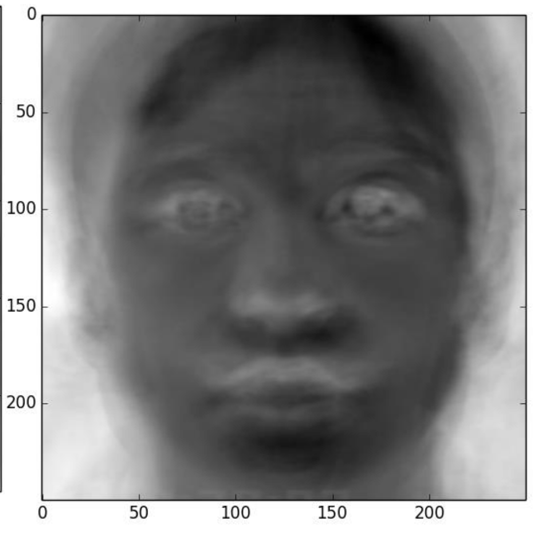
where λ_k is the k-th eigenvalue of matrix C.

Matrix V contains p columns vectors, each of length p, which represents the p eigenvectors of the covariance matrix C.

Eigenvalues and eigenvectors are ordered and paired. The i-th eigenvalue corresponds to the i-th eigenvector.

First four eigenvectors has been transformed after PCA from a single – dimensional vector to a 2D matrix and then showed as an image. They can be seen below:





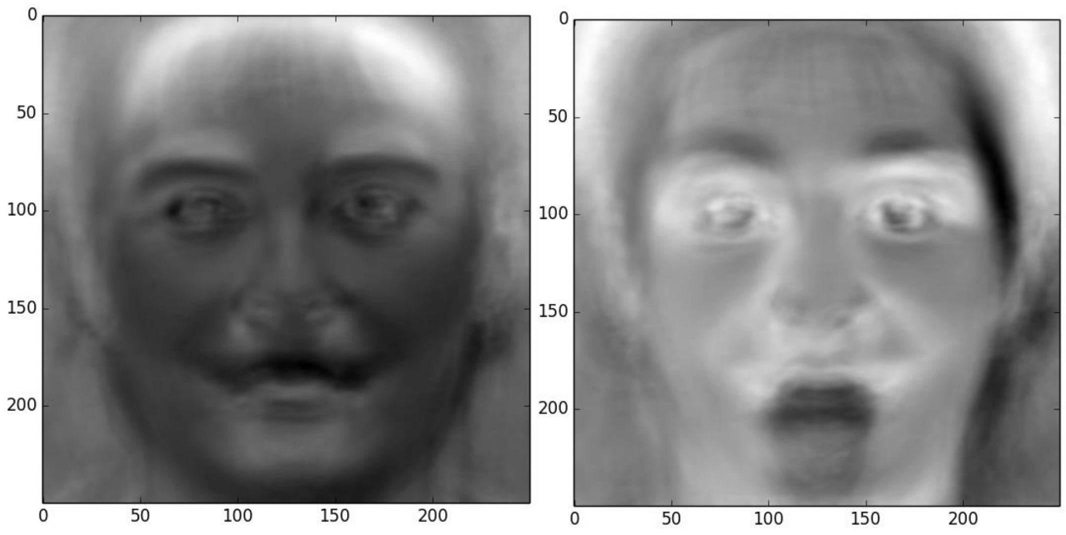


Figure 6. First four eigenvectors showed as pictures

Also the mean vector *U* can be represented as in image:

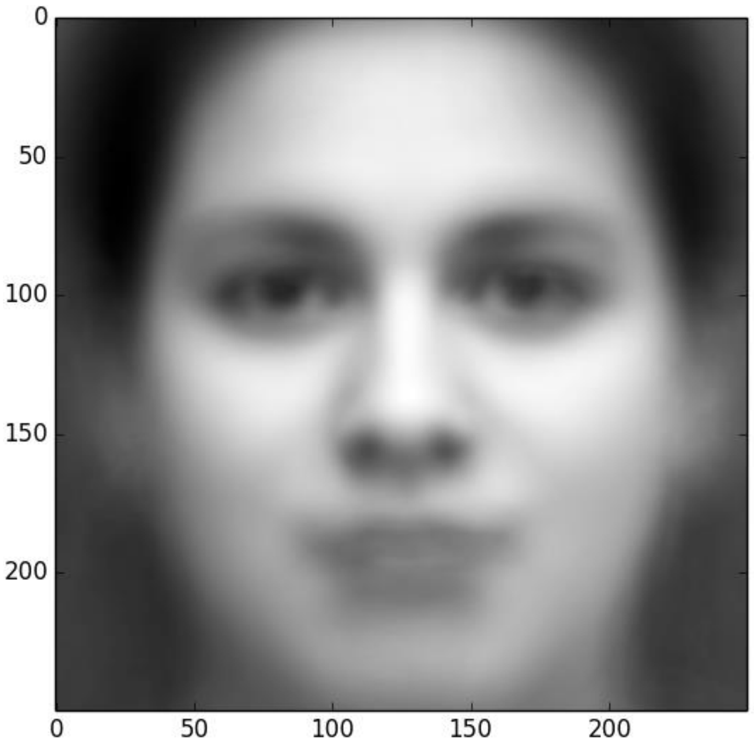


Figure 7. Mean vector as in image

For reducing the number of input features for learning the face expression recognition algorithm the number of eigenvectors has been reduced to 100. As mentioned at the beginning, the first eigenvector shows the direction in which the data vary the most. The second shows the second direction and so on. According to this, the higher the number of the eigenvector is, the less differences between the data it shows. Choosing 100 eigenvectors for further processing should not decrease the face expression prediction accuracy. After choosing the number of eigenvectors, the input matrix of 1200 pictures, after subtraction the mean value from every row, will be multiplied by an eigenvector matrix. The formula for that can be seen below:

$$Y[i] = V \cdot (X[i] - U) \tag{9}$$

where X[i] is a single vector with 62500 columns which represents all pixels of given photo. After this operation we will get a matrix Y of dimensions 1200 x 100, which presents 1200 pictures with 100 features in which the data vary the most. This data can be finally applied for learning the algorithm for face expression prediction. For future processing these photos, now each reduced to a single – dimension vector of 100 values, will be saved to binary file with corresponding emotions labels.

5. Neural Network

Artificial neural network is a computing system inspired by biological neural network which represents animal brain. These systems learn tasks (progressively improving performance) considering given examples. An example can be a task of identifying images which contain cats. It can be done by analyzing images, manually labeled as ‘cat’ or ‘no cat’, and using the results to identify cats in other images. Neural network will find these pictures in different way than human will. They do not know that cats have fur, tails, whiskers and cat – like faces. Instead of that the will extract their own set of relevant characteristic features from the learning material.

For face expression recognition a Multi-Layer Perceptron Neural Network will be used and then it will be trained using supervised learning method. MLP is an algorithm which learns a function (3.1) by training on dataset.

$$(\cdot) : R^m \to R^o \tag{3.10}$$

Where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features $X = x_1, x_2, x_3, \dots, x_m$ and a target y, it can learn a non – linear function approximator for classification. Between input and output layer there can exist one or more layers, called hidden layers. The picture below shows a one hidden layer MLP with scalar output.

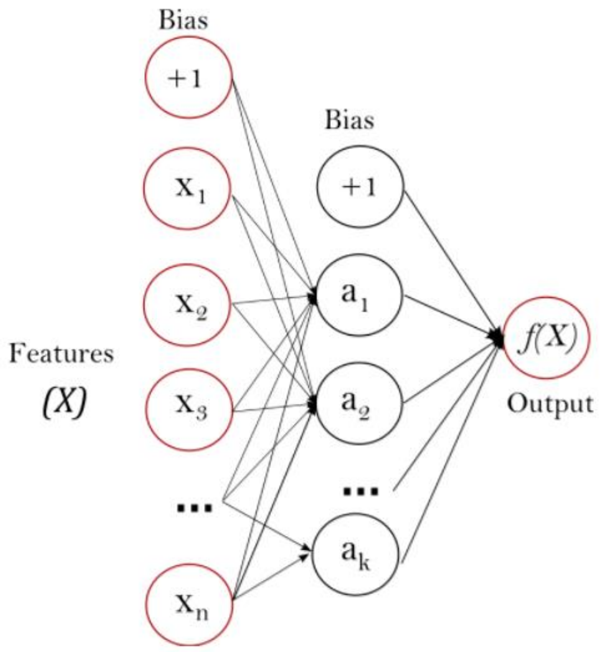


Figure 8. MLP with one hidden layer [http://scikit-learn.org/stable/modules/neural_networks_supervised.html]

The first layer, called input layer, consists of a set of neurons $\{x_1, x_2, \dots, x_m\}$ representing the input features. Neurons in the hidden layer transform the values from the previous layer with a weighted linear summation $w_1x_1 + w_2x_2 + \cdots + w_mx_m$, followed by a non – linear activation function:

$$g(\cdot) : R \to R \tag{3.2}$$

The bias is a constant value (typically one) appended to each pre – output layer. The output layer receives the values from the last hidden layer and transform them into output values.

In this project the number of input features is equal 100, what is explained in the last section of Feature extraction chapter. Previously saved pictures can now be easily accessed by opening the binary file. For choosing the best configuration of the hidden layer the loaded data need to be shuffled and split into training and testing set.

The input features are well prepared at this point but the output values are now a string labels. In the dataset we can distinguish 8 different emotions: *Contempt*, *Disgust*, *Fear*, *Happy*, *Sadness*, *Surprise*, *Anger and Neutral.* These data cannot be simply applied to train an MLP, we need to transform them firstly. The best way to do this it to use a method called *one – hot – code.* The transformation rely on writing each class label with only one '1' on given position and '0' on remained positions. For doing this for 8 different class we will need 8 bits. The output value of each picture will change, according to the pattern given below:

Emotion	One – hot – code
Contempt	0, 0, 0, 0, 0, 0, 0, 1
Disgust	0, 0, 0, 0, 0, 0, 1, 0
Fear	0, 0, 0, 0, 0, 1, 0, 0
Happy	0, 0, 0, 0, 1, 0, 0, 0
Sadness	0, 0, 0, 1, 0, 0, 0, 0
Surprise	0, 0, 1, 0, 0, 0, 0, 0
Anger	0, 1, 0, 0, 0, 0, 0, 0
Neutral	1, 0, 0, 0, 0, 0, 0, 0

Table 1. Class labels transformation into one - hot - code

The output layer will consists of 8 perceptrons, according to 8 different emotion labels. For training the MLP only a part of all available samples will be used. The rest will serve as a testing set. The whole dataset was split for this purpose into two parts. The training set includes 80% of images and testing set the remained 20%. The size of hidden layer will be selected according to results of an experiment. The configuration of MLP will start with the size of hidden layer equal to 100/10 = 10.
For this configuration the MLP predicting accuracy will be measured for the unseen photos from the testing set. After that, the number of hidden layer will be increased by multiplying the initial value by i value, which is an iterator in loop in range of 0 – 10. According to this the hidden layer size will be increased each time by 10 and will reach 100 at the end. For each size the accuracy will be measured and this will allow to choose the best configuration. The results will be shown in the next chapter.

6. Final configuration and real – time face expression detection

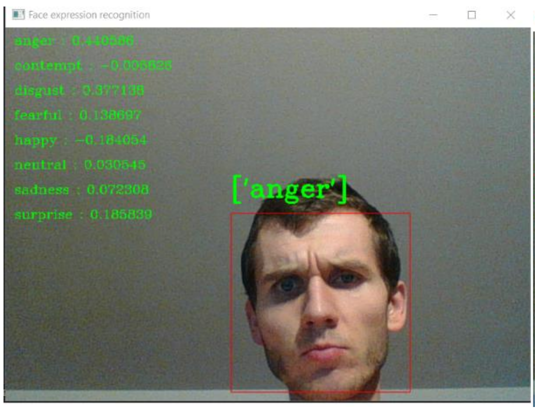
The experiment results can be seen in table below. The best accuracy has been reached with 30 perceptrons in the hidden layer.

Input layer	Hidden layer	Output layer	Accuracy
100	10	8	0,86
100	20	8	0,88
100	30	8	0,90
100	40	8	0,88
100	50	8	0,89
100	60	8	0,89
100	70	8	0,88
100	80	8	0,88
100	90	8	0,86
100	100	8	0,89

Table 2. Accuracy in dependence on hidden layer size

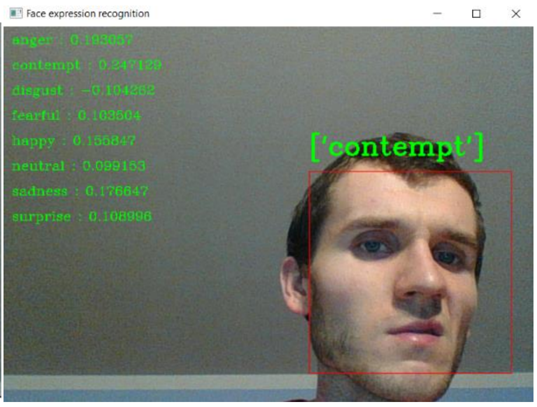
The weights, returned as a result of training the MLP was saved to a file for the real – time face expression recognition.

Using the wxPython an application for real – time face expression recognition has been developed. It allows to predict user's face expression from the images fetched from the webcam, inbuilt in every notebook. The results of testing the software by an author of this paper can be seen in the images below. The results are quite well what means that an algorithm is correct and the whole application is finished.



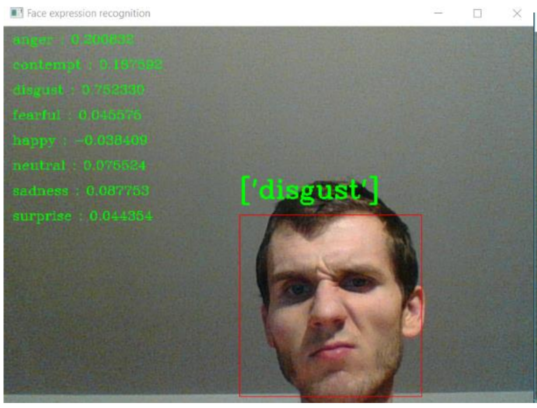
```
anger : 0.440586
contempt : -0.005828
disgust : 0.377138
fearful : 0.138697
happy : -0.184054
neutral : 0.030545
sadness : 0.072308
surprise : 0.185839
```

['anger']



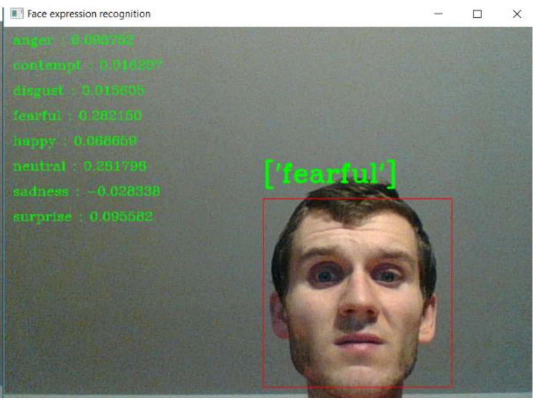
```
anger : 0.193057
contempt : 0.247129
disgust : -0.104262
fearful : 0.103504
happy : 0.155847
neutral : 0.099153
sadness : 0.176647
surprise : 0.108996
```

['contempt']



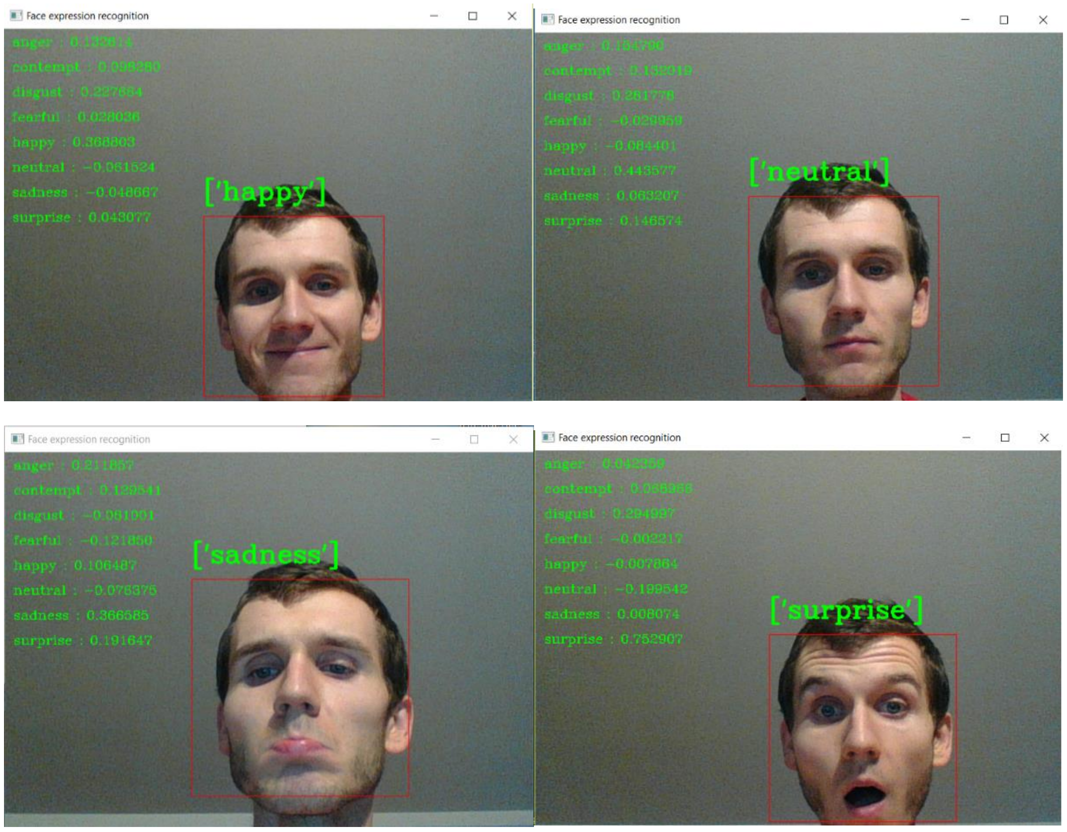
```
anger : 0.200832
contempt : 0.187592
disgust : 0.752330
fearful : 0.045576
happy : -0.038409
neutral : 0.075524
sadness : 0.087753
surprise : 0.044354
```

['disgust']



```
anger : 0.095752
contempt : 0.016207
disgust : 0.015605
fearful : 0.282150
happy : 0.068659
neutral : 0.281798
sadness : -0.028338
surprise : 0.095582
```

['fearful']



```
anger : 0.132614
contempt : 0.098280
disgust : 0.227684
fearful : 0.028036
happy : 0.368803
neutral : -0.061524
sadness : -0.048667
surprise : 0.043077
['happy']
```

```
anger : 0.154790
contempt : 0.132019
disgust : 0.281778
fearful : -0.029959
happy : -0.084401
neutral : 0.443577
sadness : 0.063207
surprise : 0.146574
['neutral']
```

```
anger : 0.211857
contempt : 0.129541
disgust : -0.081001
fearful : -0.121850
happy : 0.106487
neutral : -0.076375
sadness : 0.366585
surprise : 0.191647
['sadness']
```

```
anger : 0.042359
contempt : 0.088988
disgust : 0.294997
fearful : -0.002217
happy : -0.007864
neutral : -0.199542
sadness : 0.008074
surprise : 0.752907
['surprise']
```

Figure 9. Images presenting the finished application for real - time face expression recognition

7. References

[1] Beyeler M., *OpenCV with Python Blueprints: Design and develop advanced computer vision projects using OpenCV with Python.* Packt Publishing: 2015

[2] Seyed M., Zahir M., *Automatic facial expression recognition: feature extraction and selection.* Springer – Verlag London Limited: 2010

[3] *Dataset description and whole dataset packed as .zip file,* URL: http://www.consortium.ri.cmu.edu/data/ck/CK+/

[4] *Principal Componenet Analysis (PCA) description,* URL: https://docs.opencv.org/3.1.0/d1/dee/tutorial_introduction_to_pca.html

[5] *Multi – layer Perceptron (MLP) Neural Network description,* URL: http://scikit-learn.org/stable/modules/neural_networks_supervised.html

[6] *Artificial Neural Network description,* URL: https://en.wikipedia.org/wiki/Artificial_neural_network

[7] *Source code of GUI software,* URL: https://github.com/wizzard1994/Automatic-Face-Expression-Recognition/tree/Predict_percentages

8. List of figures

9. List of Tables