FACE RECOGNITION USING NEURAL NETWORKS

Image and Speech Recognition Project:
First report

Martin Čolja
Lovro Ludvig
Ardita Šalja

Table of contents

2. Task description

In today's modern world, image recognition is one of the most interesting and fastest developing topics; new problems arise each day. One of the most useful applications of image recognition is face recognition, which we will try to solve and implement within this project.

Furthermore, the exact task of face recognition is, given an image with a human face on it, recognize the person on the image. There are various methods with which this problem can be solved, but in this project we will be using neural networks for classification proposed in. [1]

First we will extract features based on eigenfaces and eigenvalues in order to have a proper input to our network (further explained in section 2). Then we will train our network with prepared dataset of images containing faces that we want to recognize. As an output we expect the name of the person that our network recognized.

In section 2 we will explain the feature extraction and the use of eigenfaces. Section 3 will contain basic information on how a neural network and training process work, as well as how exactly we plan to build our network. In section 4 we will present and explain our chosen dataset.

2. Feature extraction

2.1. The need for Feature extraction

A common way of classifying data, using neural networks, is to feed the input neurons of the network with each pixel value and create a network deep enough that it can extract the features itself. That method, successful as it may be, simply isn't fit for the task at hand, since it requires building a deep neural network, which is computationally heavy and requires an abundant dataset.

In order to build a smaller network, it is necessary to reduce the number of input variables. Possibilities are to try and extract individual features from an image, or to reduce the image dimensionality. There are multiple ways of doing both, however, dimensionality reduction seems to be more suited for the problem at hand.

2.2. Eigenfaces

The goal of dimensionality reduction should be to keep the divide between features as close as possible to the original image, for example, principal component analysis. That method consists of generating eigenvectors, vectors with the property of preserving direction after linear transformations, a property which will in term allow us to determine the most lucrative features of an image. [3] After the principal component analysis is completed, the resulting eigenvectors are called eigenfaces. Eigenfaces don't correspond to any facial features but are a ghostly representation of the most prominent differences between any face and an average face. [4]

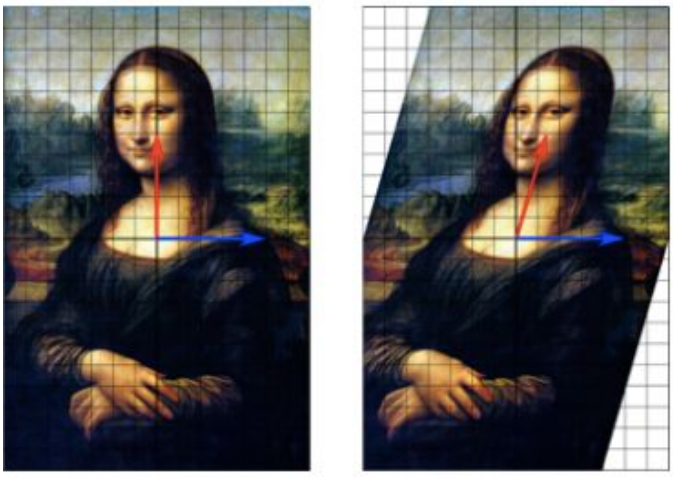

Image 2.1: Example of eigenvector maintaining direction

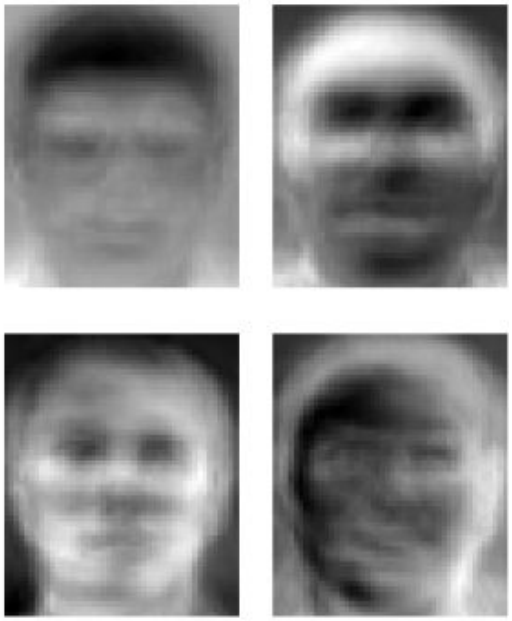

Image 2.2: Some eigenfaces from AT&T Laboratories Cambridge

2.3. Algorithm

For the construction of the eigenface, a considerate number of samples is still required. The images in question should be a good representation of faces, so once averaged, they can represent an average face reasonably well. After the images are collected, each image is, for each row, scanned from left to right and collected in a way that an entire image is now represented by a single vector. All of these image-vectors are then stored as a matrix, such that each row represents one image, and each column represents all different image values for an individual pixel. After obtaining a representation of the data, the average face can be calculated as follows [1][4]

$$\Psi = \frac{1}{N} \sum_{i=1}^{N} X_i$$

Where N is the number of images, while X represents individual pixel values for each row. The average face is subtracted from every other face, creating a new set of images. What is left are features that best differentiate that face from the average and should vary from person to person. [1][4]

$$\theta_i = X_i - \Psi$$

Using the new set of differences, we can calculate covariance matrix C using the matrix of acquired differences and its transpose. [1][4]

$$C = \theta^T \cdot \theta$$

It is further possible to calculate the N eigenvectors with corresponding eigenvalues, by finding which vectors satisfy the eigenvector condition. [1][4]

$$(C - \lambda I) \cdot \vec{v} = 0$$

With $\vec{v}$ being the resulting eigenvector and λ being the corresponding eigenvalue for the vector. The resulting eigenvectors are called eigenfaces, and their sum represents the original image; however, not all of them are required to give a good representation of the initial face. [1][4]

Eigenvalues determine how much of an impact a particular eigenvector has in representing an image. Since eigenvalues tend to drop rather quickly with the increase of eigenfaces, it is possible to represent a face with a significantly smaller number of eigenfaces than their total amount. What is left is to select the best eigenvalues and to make a matrix U with their corresponding eigenvectors. The resulting matrix is actually a transformation that will transform any new image Z into a point in eigenface space. The coordinates of an image in that space, the face vector ω, serve as inputs for the neural network. [1][4]

$$\omega = (Z - \Psi) \cdot U^T$$

3. Using a neural network for classification

3.1. Architecture of the neural network

After a specific choice of the features is chosen, it is time to do the classification. For this project, we will use a 3 layer neural network (image 3.1) for the classification task. Layer 1 will be input for Eigenfaces, layer 2 is the hidden layer and layer 3 is the output layer.

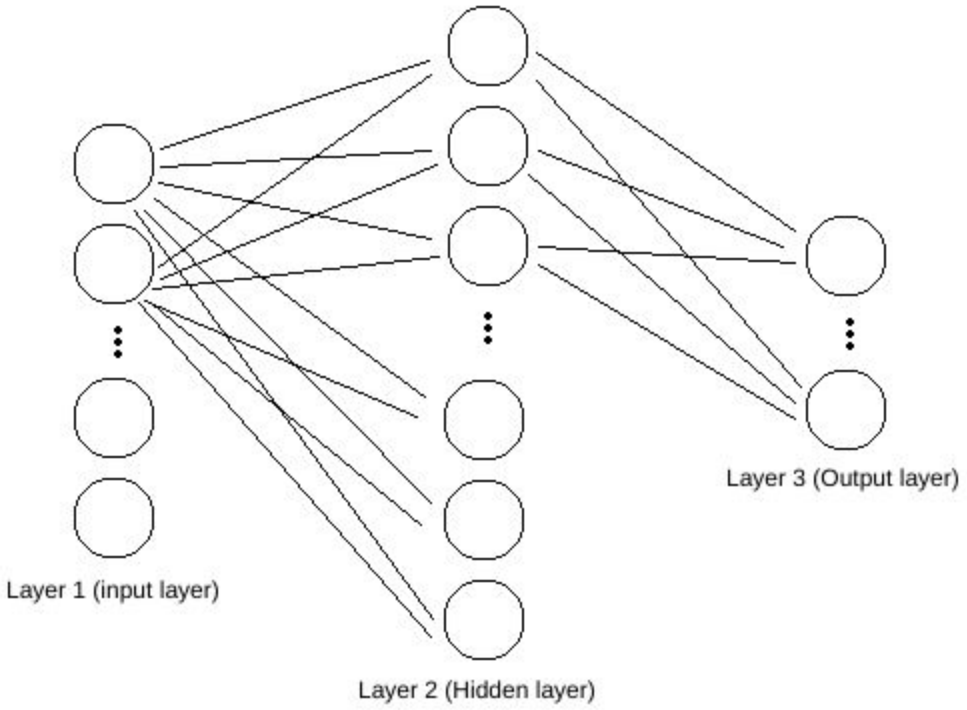

Image 3.1: 3 layer Neural network

Each layer consists of a number of perceptrons (Image 3.2) where each perceptron is connected to all perceptrons from the previous layer. This kind of architecture is called The Multilayer Perceptron (MLP). [6]

While only one perceptron allows linear decision boundaries, having more perceptrons in a cascade and creating an MLP allows as to create complex boundaries. When perceptrons are put like that in a neural network, we can start calling them neurons as they act like one. For a neuron, it is also important to define an activation function. We will use a sigmoid function (Image 3.3) as it is one of the most used functions in MLP. It has the advantage of giving values between 0 and 1 and that allows us to know what is the probability of each class. [6]

For our neural network, the number of neurons in the first layer will depend on the number of chosen Eigenfaces (expected to be around 10-30, exact number will be determined while implementing the algorithm). In the second layer, we should put as many neurons as possible without overfitting the neural network. The number of neurons in the third layer depends on a number of classes being classified, we chose to have 7 (5 classes for faces we classify, 1 class for unknown faces, 1 class for background).

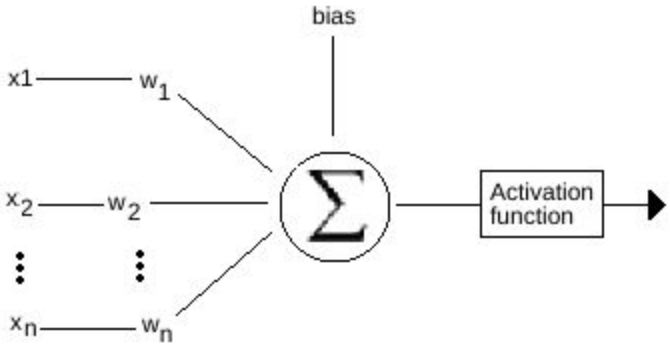

Image 3.2: Perceptron

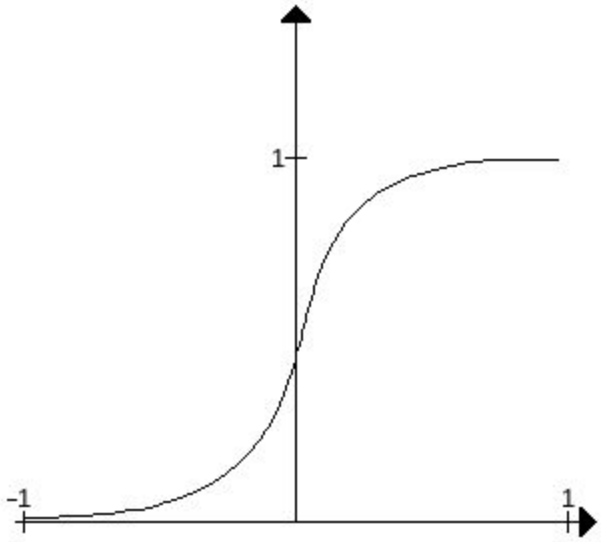

Image 3.3: Sigmoid function

3.2 Training the neural network

The backpropagation algorithm is based on gradient descent learning rule and is the most used learning algorithm for training MLP. It is called backpropagation because the input sample is first propagated through the network in a forward direction after which it is propagated back to make corrections on neuron weights. It does corrections first by calculating the error and then going back with differentiable activation function units. Backpropagation algorithm follows: [5][7]

Repeat until the stopping criterion is satisfied:

For each training example s : $(x_{s,1},\ldots,x_{s,N_i}) \rightarrow (t_{s,1},\ldots,t_s,N_o)$ *do:*

1. Set the example $(x_{s,1},\ldots,x_{s,N_i})$ *as the input of the network.*

2. Compute outputs of all neurons from all layers, from the first to the last layer; denote the outputs of the final layer as $(o_{s,1},\ldots,o_{s,N_o})$.

3. Determine errors of output layer neurons:

$$\delta_i^K = o_{s,i}\cdot(1-o_{s,i})\cdot(t_{s,i}-o_{s,i})$$

4. Go back layer by layer toward the first layer. For the i-th neuron of the k-th layer the error is:

$$\delta_i^{(k)} = y_i^{(k)}\cdot(1-y_i^{(k)})\cdot\sum_{d\in Downstream} w_{i,d}\cdot\delta_d^{(k+1)}$$

5. Modify all weights. Weight $w_{i,j}^{(k)}$ *is modified as:*

$$w_{i,j}^{(k)} \leftarrow w_{i,j}^{(k)} + \eta\cdot y_i^{(k)}\cdot\delta_j^{(k+1)}$$

and the thresholds are modified as:

$$w_{0,j}^{(k)} \leftarrow w_{0,j}^{(k)} + \eta\cdot\delta_j^{(k+1)}$$

4. Dataset

In this project we will be using an existing Collection of Facial Images: Faces95 which contains 20 images per individual and there is 72 individuals. All images are in RGB and have resolution of 180x200 pixels. On images individuals are shown in front of a background representing a red curtain.

Since we want our network to be able to recognize 5 different persons, we will be using 15 pictures for each distinct face that we want to recognize in order to train our network. In training we will also use 10 pictures of unknown people and 5 images of background. In total that is 90 training images.

For testing we will be using 5 images for each person that was in training set. We will also be using 5 pictures of faces that were not initially in training set. In addition we will be using 5 pictures of background. To sum up, there will be 35 testing images in total.

Image 3.4. Example of images in our dataset [2]

Image 3.5. Example of background image [8]

5. Bibliography

[1] N. Jamil, S. Lqbal, N. Iqba: Face recognition using neural networks, IEEE, https://ieeexplore.ieee.org/document/995351?arnumber=995351, 2002.

[2] Collection of Facial Images: Faces95, URL https://cswww.essex.ac.uk/mv/allfaces/faces95.html

[3] 3Blue1Brown Eigenvectors and eigenvalues | Essence of linear algebra, chapter 14 https://www.youtube.com/watch?v=PFDu9oVAE-g&t=419s

[4] I. U. Wahyu Mulyono, D. R. Ignatius Moses Setiadi, A. Susanto, E. H. Rachmawanto, A. Fahmi and Muljono, "Performance Analysis of Face Recognition using Eigenface Approach," 2019 International Seminar on Application for Technology of Information and Communication (iSemantic), Semarang, Indonesia, 2019, pp. 1-5. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8884225&isnumber=8884215

[5] Bojana Dalbelo Bašić, Marko Čupić, Jan Šnajder, Neural networks, FER, Zagreb, https://www.fer.unizg.hr/_download/repository/UI_12_UmjetneNeuronskeMreze[1].pdf

[6] D. R. Hush and B. G. Horne, "Progress in supervised neural networks," in IEEE Signal Processing Magazine, https://ieeexplore.ieee.org/document/180705

[7] P Latha, L Ganesan, S Annadurai - Face Recognition using Neural Networks, http://www.cscjournals.org/download/issuearchive/SPIJ/Volume3/SPIJ_V3_I5.pdf#page=75

[8] HaarTraining: Background images, URL https://github.com/sonots/tutorial-haartraining