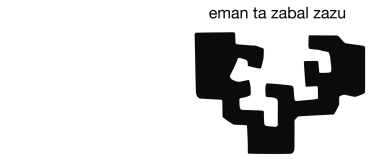


Grade in Informatics Engineering

Computer Science

Bachelor Thesis

Analysis of facial expressions in children: Experiments based on the DB Child Affective Facial Expression (CAFE)

Author

Leire Roa Barco

2016

Contents

List of Figures

List of Tables

1. CHAPTER

Introduction

Humans can recognize facial expressions and trough them can deduce emotions. However there are some human beings, like autistic persons, who can not deduce well emotions by reading the human facial expressions. The aim of this paper is to use the computer as a tool which can recognize the facial expresions of children between 4 and 8 years old and deduce their emotions. It could be used in the future as a way to improve the emotional intelligence of autistic children and persons with similar difficulties. Computer vision researchers, who work in facial expression recognition, use machine learning systems for taking large sets of appearance-features as input, and train the computer program on a large database of examples.

In order to make it, we are going tu use Convolutional Neural Networks, but what are CNN or ConvNet? To understand this, first we need to understand the (Artificial) Neural Networks (ANNs). Neural networks are multi-class classifiers that are inspired by the biological neural networks (the central nervous systems of animals, in particular the brain). This classifiers are repressented as systems of interconneceted "neurons" which exchange messages between each other, as seen in Chapter 2. Usually to function they require huge amounts of training data, they are computationally intensive to train and many times it is hard to optimise. CNNs are really similar to this ones, they are also biological neural networks, but in this case they use other kind of units as "neurons" called perceptrons so that the use the minimal amounts of preprocessing, and are a type of neural networks called Feed Fordward Neural Network. They have an implementation in Matlab called Neural Network Toolbox [Demuth and Beale, 1993], but there is also another toolbox which is not the official one of Matlab and it is open source, it is called MatConvNet

[Vedaldi and Lenc, 2015], and due to any of these implementations, Matlab can obtain the features of an image. Check Chapter 3 for more details.

As mentioned previously each one of the images has some features, and there are different ways of detecting them, as we can see in the Chapter 4 where we can read about the state of the art of the facial expression recognition. Nevertheless, in order to do any kind of experiment, it is necessary to work with a database, and in our case we used a database of children [LoBue, 2014], that needed some preprocessing before being able to use as it is shown in Chapter 5. Finally, all the done experiments and the conclusions of them are shown in the chapter 6.

2. CHAPTER

Theoretical Neural Networks

A neural network is a classifier that has inputs and output; it is a classifier made by connected neurons in which every neuron calculates when it has to go on or go off depending on the threshold, and as well as with the classifiers, neural networks are trained too, and it is inspired by biological neural networks

2.1 Perceptrons

The "neurons" that compose any Neural Network can be of different kinds, and one of the most basic ones are the perceptrons. This "neurons" were developed in the 1950s and 1960s by the scientis Frank Rosenblatt based on the work by Warren McCulloch and Walter Pitts.

Perceptrons can be understood in two different ways: as a neuron or as a classifier. If we understand the perceptron as a classifier, it is a type of linear classifier that takes several binary inputs x_1, x_2, ..., and produces a single binary output.

These elements have to one or more inputs x_1, x_2, ..., x_n, where each of the inputs have a weight w_1, w_2, ..., w_n, and these weights are used to determine the output of the "neuron" which is a single binary value. If a perceptron has an output but it has not inputs the perceptron would simply output a fixed value, not the desired value. So, it is better to interpret the input perceptrons not as perceptrons themselves, but as units defined to

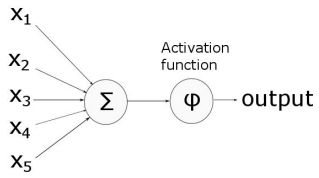


Figure 2.1: Perceptron

output the desired values. The perceptron which can be understand as an algorithm for learning a binary classifier whose function can be defined as follows:

$$\mathrm{f}(x) = \begin{cases} 1 & \text{if w} \cdot \mathrm{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Where w is the vector of the weights, x is the input real value vector and $w \cdot x$ is the dot product $\sum_{i=0}^{n} w_i x_i$ being n the number of inputs and b the bias. This bias shifts the decision boundary away from the origin and does not depend on any input value. This function basically describes an hyperplane that separates the two regions we want to classify.

One problem with the perceptrons is that the perceptron learning algorithm does not determinate if the learning set is linearly separable, and because of this if the vectors are not linearly separable it will never reach a point where all the elements are classified. An example of this problem is the Boolean exclusive-or problem (XOR). However, according to Rosenblatt's theory, a Perceptron can also be understand as a neural network of perceptron (a multi-layer perceptron which will be explained later) and the perceptron algorithm is interpreted as a single-layer perceptron (which is the simplest feedfordward neural network). This concept of multilayer network perceptron (MLP) is able to implement all the logic arithmetic operators such as AND, OR, XAND. among others, that can lead to sophisticated decision making.

2.2 Sigmoid neurons

If we make a small change in the inputs of the perceptrons understanding them as functions, the output might be highly affected by this. The sigmoid neurons, also known as sigmoid functions, are the solution to this: They are a really similar classification function, but instead they can handle small changes in the input without the whole network

getting strongly affected.

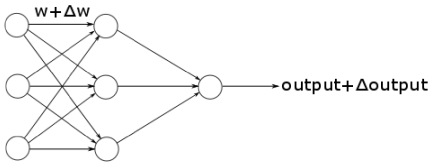


Figure 2.2: Sigmoid Neurons

The sigmoid function as the perceptron function has as an input a vector x where each of the values of the vector has its own weight of w_1, w_2, ..., w_n. However, unless in the perceptron, the output is not a 0 or a 1, it is defined by a function called sigmoid function δ (also called sigmoid activation function) defined by:

$$\delta(z) = \frac{1}{1+e^{-z}} \tag{2.1}$$

It is an S shape (sigmoid curve) function where the input z is w $z = w \cdot x + b$, being w the weight vector and x the input binary vector.

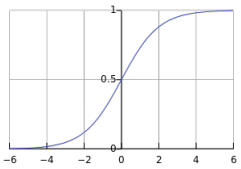


Figure 2.3: Logistic Function from [Wikimedia Commons, 2008]

As seen in the figure above, it has horizontal asymptotes in $z \to \infty$ that the value is 1 and in $z \to -\infty$. So, when the value of z is negative and small, the sigmoid function is close to the perceptron fucntion. In the other cases the functions are really different.

2.2.1 Tanh function

Another similar function to the sigmoid function is the tanh function (also called hyperboic funcition), which is nothing more than a rescaled version of the sigmoid function, whose range is [-1,1] instead of [0,1]. Usually it converges faster inside a neural network because of the derivative.

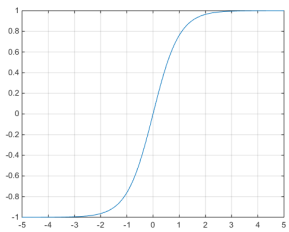


Figure 2.4: Tanh function from [apiexamples.com, 2015]

2.2.2 Rectifier

There is also a activation function close to the previously metioned ones called rectifier which is defined as

$$f(x) = \max(0,x) \tag{2.2}$$

where x is the input to a neuron. This is the most used activation function for deep neural networks and the units that use this method are called rectified liner units (ReLUs).

2.2.3 Radial Basis Function

A Radial Basis Function (RBF) $\phi(x)$ is a function with respect to the origin or to a certain point ($\phi(x)$=$\phi(\| x \|)$ and $\phi(x)$=$\phi(\| x - c \|)$ respectively), where the usual norm is the Euclidean norm, but it can be a different one. There are different kind of radial basis functions where r=$\| x - x_i \|$), the most common ones are:

- Gaussian function: $\phi(r) = e^{-(\varepsilon r)^2}$
- Multiquadratic function: $\phi(r) = \sqrt{1 + e^{\varepsilon r}}$
- Inverse multiquadratic: $\phi(r) = \frac{1}{\sqrt{1+e^{\varepsilon r}}}$

However, it is mostly used in the form of

$$y(x) = \sum_{i=1}^{m} w_i \phi(\| x - x_i \|) hwere$$

where the function y(x) is the sum of N radial functions, each one with a differenct center x_i and weighted by an coeffient w_i [Theodoridis and Koutroumbas, 2008].

2.3 Architecture of Neural Networks

2.3.1 Feedfordward Neural Networks

Feedfordward Neural Networks are acyclic graphs usually composed at least of 3 layers: The input layer, the hidden layer and the output layer.

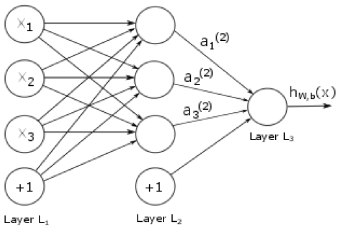


Figure 2.5: Neural Network model

If there are more than 3 layers in a Neural Network, all these extra layers are hidden layers. Every layer is composed by units $x_1, x_2 ...$, and all the layers except the output layer also have a bias units, that is the intercept term and it is usually represented with a "+1".

For example in the layer in the image we have a Neural Network composed of 3 layers, and two of them the first and second layer are composed by 3 input units and a bias unit. These bias units do not have any input connection to them, because they always output the value "+1". So that, we have a neural network $(\Theta)=(\Theta^0, \Theta^1, \Theta^2)$. The representation of this neural network is as follows:

$a_i^{(j)}$ = "activation" of unit i in layer j (a_0 represents the bias unit).

$\Theta^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j+1$.

$g(x)$ = logistic activation function.

$$
\begin{aligned}
a_1^{(2)} &= g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3) \\
a_2^{(2)} &= g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3) \\
a_3^{(2)} &= g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)
\end{aligned}
$$

$$
h_\Theta(x) = a_1^3 = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})
$$

If we want to put it in a more compact way, it can be expressed in a vectorized way:

$$z^2 = \Theta^{(1)}x \qquad z^2 = \Theta^{(1)}a^{(1)}$$
$$a^2 = g(z^{(2)}) \qquad a_0^{(2)} = 1$$
$$z^3 = \Theta^{(2)}a^{(2)}$$
$$h_\Theta(x) = a^{(3)} = g(z^{(3)})$$

This kind of Neural Networks instead of being constrained to fit the features x_1,x_2,x_3 to logistic regression, it has the flexibility to learn its own features $a_1^{(2)}$, $a_2^{(2)}$, $a_3^{(2)}$ to fit into logistic regression.

Depending on which parameters it chooses for z_1 it can learn some interesting features, and therefore we can end up with a better hypothesis than if we were constrained to use the features x_1, x_2, x_3 or contrained to use the polinomio terms $x1x2, x2x3$ [Turing Finance, 2014]. [1]

2.3.2 Other architecures of Neural Networks

Till now we have talked about acyclic directed graphs where the output from one layer is used as the input to the next layer. This kind of networks are called Feedforward Neural Networks. Nevertheless, there are other kind of artificial networks that have cycles, which are called Recurrent Neural Networks. In these models the idea is that neurons are activated for a limited time before getting deactivated. The activated neurons might stimulate other neurons to get activated later, for a limited period of time, and this can cause even more neurons to get activated, so we get kind of a cascade of activation of neurons. In these models the cycle is not a problem because the neurons are affected after a period of time, not instantaneously. There are also some kind of networks like the Radial BasissFunction (RBF) Network, which is a function that has built into it a distance criterion with respect to a center, or Modular Neural Networks like the Associative Neural Network (ASNN) where several small networks cooperate to solve problems, to mention some, but we are going to focus on Convolutional Neural Networks. However, all of this kind of Multilayer Network have one thing in common: They implement linear discriminants, but in a space where the inputs have been mapped nonlinearly, and because of that they admit

[1] If the network has s_j units in layer j, $s_j + 1$ units in layer $j + 1$, then Θ^j will be of dimension s_{j+1}x(s_j+1).

pretty simple algorithms where the form of the nonlinearity can be learned from training data.

2.4 Training of Feedfordward Neural Networks

Any multilayer Neural Network needs to be trained in order to learn the features of the model, so that it is easier for the Network to solve the classification problem we want to solve.

2.4.1 Cost Function

The cost function of a Neural Network is a measure of how good the Neural Network is performing, and learning algorithms search through the solution in order to find a function that minimizes the cost.

Having a training set $(x_{(1)},y_{(1)})$, $(x_{(2)},y_{(2)})$, ... ,$(x_{(n)},y_{(n)})$, we define the regularized logistic regression cost as follows:

$$J(\Theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\Theta(x^{(i)}) + (1-y^i)\log(1-h_\Theta(x^{(i)}))] + \frac{\lambda}{2m}\sum \frac{n}{j=1}\Theta_j^2$$

The first term of the sum represents the cost function and the second term is the regularization term (Note that we start the sumatory from j=1 beause we did not regularize the bias term Θ_0). If we are doing a binary classification with for example a tanh function, then we would have y=0 or y=1, and $h_\Theta \in \mathbb{R}$ and there would only be one layer. Whereas, if we have a multiclass classification, $y \in \mathbb{R}^k$, and we would have k outputs units, so $h_\Theta(x) \in \mathbb{R}^k$, always being $k \geq 3$.

The cost function for the whole Neural Network would be:

$$J(\Theta) = -\frac{1}{m}[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)] +$$

$$\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

As in the previous equation, the first term of the equation is the cost function itself and

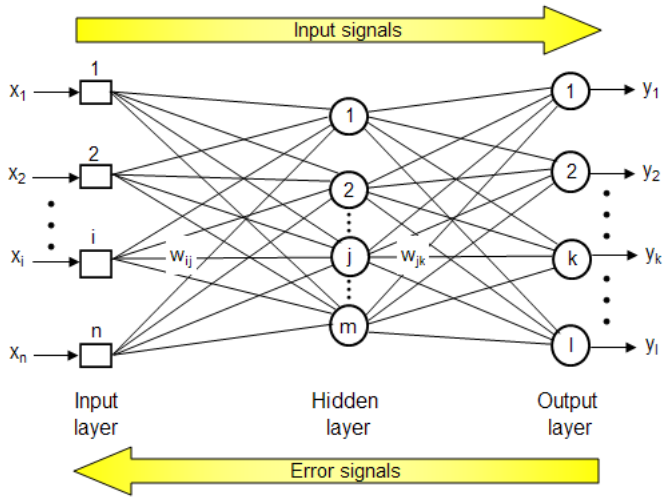


Figure 2.6: Feedforward NN with backpropagation [Buranajun et al., 2007]

the second term is the regularization term, where we do not regularize the bias terms. $h_\Theta(x) \in \mathbb{R}^k$ and $h_\Theta(x)_i = i_{th}$ output.

2.4.2 Backpropagation

One of the most popular methods for training this layers is the backpropagation algorithm, an algorithm based on gradient descent (a first order optimization algorithm that minimizes functions by iteratively moving in the negative direction of the function gradient). It is a suppervised training algorithm used for training artificial network, that takes an input and forwards it through all the layer till the last layer. Once in there it compares the expected output with the real output and an error value is calculated for each one of the outpus. This errors are backpropagated till the input layers, but this layers do not obtain all the same error value, they receive a value depending on their contribution to the total error. The idea is that once the training is done, and each time the training is done again, the neurons will learn to recognize the features of the inputs and the intermediate layers will organize themselves.

The algorithm of this process would be:

Training set $(x^{(1)},y^{(1)}), \ldots, (x^{(m)},y^{(m)})$
Set $\Delta_{i,j}^{(l)}=0$ (for all l, i, j) (used to compute $\frac{\partial}{\partial\Theta_{ij}^{(l)}}J(\Theta)$
for m times **do**
    Set $a^{(1)}=x^{(i)}$

Perform forward propagation to compute $a^{(l)}$ for $l = 2,3,\ldots,L$
Using $y(i)$, compute $\delta^{(L)}$=$a^{(L)}$-$y^{(i)}$
Compute $\delta^{(L-1)}$,$\delta^{(L-2)}$,...,$\delta^{(2)}$
Use backpropagation $\Delta_{ij}^{(l)}$:=$\Delta_{ij}^{(l)}$+$\delta_i^{(l+1)}(a^{(l)})^T$
end for
if $j \neq 0$ **then**
$D_{ij}^{(l)}$:=$\frac{1}{m}\Delta_{ij}^{(l)}$ + $\lambda\Theta_{ij}^{(}l)$
else
$D_{ij}^{(l)}$:=$\frac{1}{m}\Delta_{ij}^{(l)}$
end if

In the algorithm, first of all we have a training set of m elements. Then the deltas are set to 0 because these are going to be used as acumulators to compute the partial derivatives. Then we loop through the whole training set, and when we are in the i=1 that means we will be working with the sample (x^i, y^i). First of all we set the activations of the input layer to equal x^i, then we compute propagation to the rest of the layers till the last layer, layer L. After that y^i is used to compute the error $\delta^{(L)}$ for the output layer (it is the hypotesis output $a^(L)$ - what the target label was). Then, with backpropagation we are going to calculate $\delta^{(L-1)}$, $\delta^{(L-2)}$, ..., δ^2 (There is no δ^1 because we do not associate the error term with the input layer). Finally we use the delta terms to accumulate the derivative terms. At the end we compute $D_{i,j}$, where if j=0 it means that that is a bias term. And, that is the partial derivative of the cost function to each of the parameters, and that can be used in gradient descent or other optimization algorithms.

Depending on how it is implemented we might end up calculating δ also.[2] To solve it, we can use gradient checking and implement $\frac{d}{d\theta}J(\theta) \approx \frac{J(\theta+\varepsilon)-J(\theta-\varepsilon)}{2\varepsilon}$ [Yu et al., 2015] [Huang et al., 2015][Zhous et al.,].

2.4.3 Other training algorithms

The most popular and most used algorithm is the backpropagation algorithm, however, there are also other training algorithms, which, unless the backpropagation algorithm, are useful to optimize both, the architecture and weights of neural networks.

One of them is the Particle Swarm Optimization (PSO), where we construct a population

[2]There is no $\delta^{(1)}$ because the first layer corresponds to the input layer and that is just the feature we observed in our trainning set, and it does not have any errors associated with it.

called swarm of candidate solutions (particles). These solutions are moved according to the best know position in the search-space as well as the entire swarm's best knowng position. Whenever improved places are found, they will guide the movements of the swarm. This process is repeated iteratively, however, it is not guaranteed that in each iteration a learning is going to occur, and it is neither sure whether a propper solution is going to be found.

Another algorithm is the Genetic Algorithm (GA). In this algorithm a population of vector represented neural networks is constructed. Then 3 steps are done: Selection, the top percentage of the population are selected using the sum-squared error of each network; Crossover, a child solution is created in each offspring with weights from both 'parent' neural networks; and finally, Mutation, to maintain the diversity in the population, a small percentage of the population are selected to go under mutation.

2.4.4 Random Initialization

In order to work with a Neural Network it is neccesary to initialize the input Θ so that the Network starts to learn. One option might be to initialize the Θ to zeros, however, if we do this, after each update all the hidden layers are computing the exact some features, and this prevents the Neural Network form learning something interesting. So, it would be a better idea to intialize each $\Theta_{ij}^{(l)}$ to a radom value in $[-\varepsilon, \varepsilon]$. This process is called symetry breaking. However, this still fails to break the symetry in a full neural network, so we need to do the following steps:

- Implement backpropagation
- Do gradient checking
- Use gradient descent or one of the advanced optimization algorithims to minimize $J(\Theta)$ for the parameters starting from this randomly initialized parameters (symetry breaking)

Hopefully it will be able to find a good value of Θ.

3. CHAPTER

Convolutional Neural Networks

A Convolutional Neural Network is a kind of neural network where the weights are shared: all the neurons in the hidden layer share the same parametrization (the weight vector and the bias, forming a feature map are the same for each pixel in the layer), and because of this the gradient of a shared weight is the sum of the gradients of the parameters being shared. Since some parameters are shared the total number of parameters is reduced drastically in comparison with other Neural Networks. It uses small convolution kernels to search across the input image for specific visual features, and it uses a neural-network back-end to interpret the patterns of features that emerge from the convolutional layers. It also uses the back-propagation algorithm, that is not only used for trainning the neuron connection, it is also utilized to train the convolution kernels. Due to all the previously mentioned features, when a CNN classifies, instead of diving the classification plane with straight lines, it is able to divide it with curve lines.

In studies of classifiers for high dimensional image data Convolutional Neural Networks (CNNs) have become the representatives among other deep learning methods. Until the stochastic diagonal Levenberg-Marquardt algorithm for CNN proposed by LeCun in 1998 [LeCun and Yann, 1998] there were not efficient training algorithms. Handwritten character recognition, natural image processing, etc, are famous engineering application of CNNs.

A interesting property of convolutional layers is that if the input image is shifted, the feature map output will be shifted by the same amount, but will be left unchanged otherwise, so that the convolutional networks are robust against shifts and distorsions of the

input, as shown in the book [Sun and Jin, 2016], where they used a Convolutional Neural Network stochastic gradient descent with I2 regularization for human facial expression classification, and the experiment showed that this networks have the capacity to clasify expression images with transational distortion. The weight sharing technique has also an interesting side effect: it reduces the number of free parameters, reducing the "capacity" of the machine and reducing the gap between test error and training error.

3.1 Max Pooling

Max Pooling is a form of non-linear down-sampling that divides the input image into a set of non-overlapping rectangles and for each such sub-region, outputs the maximum value, that then can be used for classification. By doing this, the computation is reduced for upper layers and it provides a form of translation invariance; by adding robustness to position, max-pooling reduces the dimesion of intermediate representations.

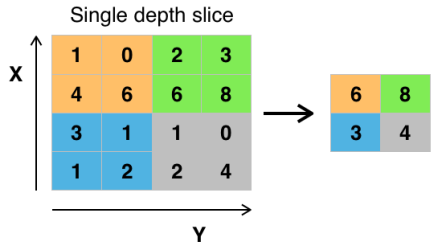


Figure 3.1: Maximum pooling from [Wikimedia Commons, 2015]

The typical values are 2x2, however, for very large images may be necessary to use 4x4 pooling in the lower-layers. Nevertheless, we might be careful because it reduces the dimension of the signal by a factor of 16, and may be taking rid of too much information.

There is also another kind of pooling called average pooling, where in every sub-region the average of the values are taken (See Figure 6).

3.2 Structure

One typical case of Convolutional Neural Networks, is LeNet-5 that is composed by layer called C1, S2, C3, S4, C5 and F6, where the C1 and C3 are convolution layer, S2 and S4 are subsampling layers and F6 is the output feature map.

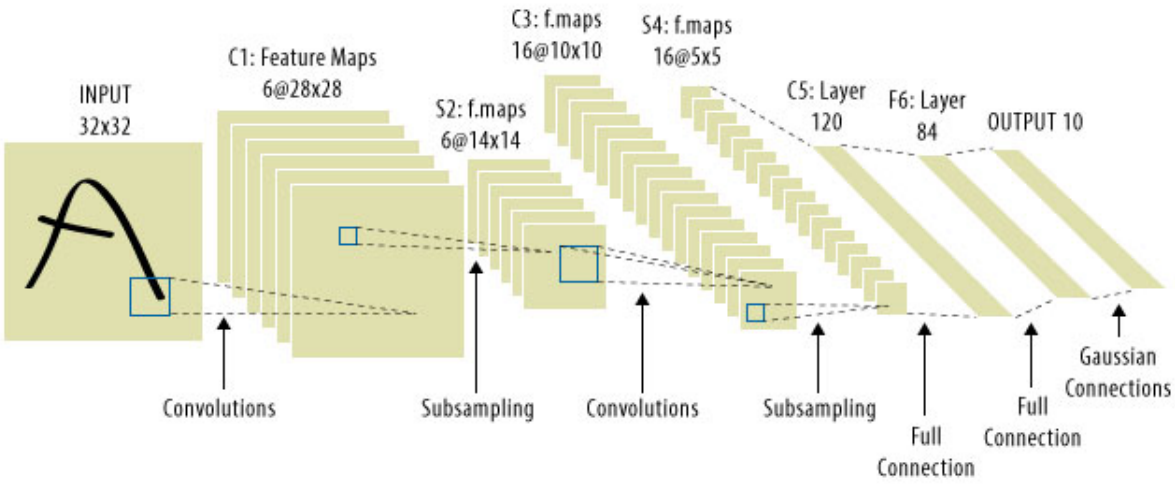


Figure 3.2: Usual structure of a CNN [LeCun and Yann, 1998]

The convolution layer is a layer that uses the convolutional kernel throught the pixels in the layer and gives as a result a bit smaller frame in which the value of each pixel expresses how closely the cell cornered at the pixel resembled the feature in the convolution kernel. This resulting figure might have different shapes. This is due to the kernel which is a small matrix that depending on the values can cause different effects like sharpening and edge detecion among others.

For every of the layer in the CNN, this convolution process will be repeated many times, each times with a different kernel, and this results will create what we call a feature map. Each of this maps will have nearly the same number of pixels as the original image, but the processing needed for this is really small compared with what a neural layer would have required.

After all of that, the convolutional layer applies a de-linearization to make sure that the outputs are not a linear combination of the inputs.

An example of a convolution is the next one, where a photo is processed by a 3x3 kernel (the second matrix) and creates 4 feature layers.

If a CNN has more than one convolutional layer, the deeper layers will have multiple feature maps, as the first one does, but this feature maps will come from 3D rather than 2D convolutions. That means, that each pixel in the feature map will be a weighted sum of a small 3D kernel times the pixels in the corresponding cells from each of the feature maps in the preceding layer.

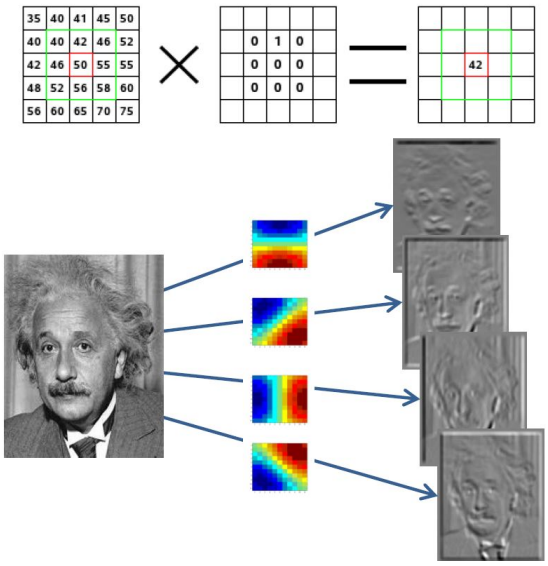


Figure 3.3: Example of convolution in a photo [Raj, 2016]

Sub-sampling layers on the other hand, are really different to convolutional layers, they do a local averaging and sub-sampling so that it reduces the resolution of the feature map, and reduces the sensitivity of the output shifts.

The structure of the inside of the LeNet-5 Neural Network and their use with the gradient based learning is the following one:

First input image is transformed by 3-dimensional convolution with six 5*5*1 sized kernels, added by bias term, activated by tanh function. After that, they get the first set of six feature maps called C1.In this layer, each unit in each feature map is connected to a 5x5 neighborhood in the input. The size of the feature map is 28x28 which prevents connection from the input from falling off the boundary. C1 contains 156 trainable parameters, and 122,304 conections.

Second, max-polling proccess is applied and S2 layer will be get. Layer S2 is a sub-sampling layer with 6 feature maps of size 14x14. Each unit in each feature map is connected to a 2x2 neighborhood in the corresponding feature map in C1. The four inputs to a unit in S2 are added, then multiplied by a trainable coefficient, and added to a trainable bias. The result is passed throug H a sigmoidal function. The 2x2 receptive fields are non-overlapping, therefore feature maps in S2 have half the number of rows and column as feature maps in C1. Layer S2 has 12 trainable parameters and 5,880 connections.

Third, the layers C3 and C4 are created with the exactly same mecanism. Layer C3 is a convolutional layer with 16 feature maps. Each unit in each feature map is connected to

several 5x5 neighborhoods at identical locations in a subset of S2's feature maps. Layer C3 has 1,516 trainable parameters and 151,600 connections. Whereas layer S4 is a sub-sampling layer with 16 feature maps of size 5x5. Each unit in each feature map is connected to a 2x2 neighborhood in the corresponding feature map in C3 in a similar way as C1 and S3. Layer S4 has 32 trainable parameters and 2,000 connections.

Fourth, two fully connected layers called C5 and F6 are calculated by the same way as the conventional neural networks. Layer C5 is a convolutional layer with 120 feature maps. Each unit is connected to a 5x5 neighborhood on all 16 of S4's feature maps. There is a full connection between S4 and C5 because the size of S4 is also 5x5 and the size of C5's feature map is 1x1. Layer C5 has 48,120 trainable connections. Layer F6 on the other hand contains 84 units and is fully connected to C5. It has 10,164 trainable parameters. [LeCun and Yann, 1998][Sun and Jin, 2016].

Finally in order to enable the classifer to reject unreasonable inputs a Gaussian layer is used to compute the distance between 84-dimension activation data of F6 and 10 fixed binary codes. Training progress could be understood as a fitting job with respect to F6 in order to minimize the Gaussian distance between F6 layer′s activation and its nearest binary code.

As in classical neural networks, units in layers up to F6 compute a dot product between their input vector and their weight vector, to which bias is added.

If we consider x0 as the inputs and fL as the loss function, an L - 1 layered CNN can be considered as a system built by a cascade of transformation modules f1(x;w),f2(x;w),..., fL(x;w) whose inputs and outputs are connected one after another. Each transformation is differentiable, thus the gradient-based learning algorithm can be employed for training CNNs.

Sun and Jin use the Alex-Net and Zeiler′s Net [Sun and Jin, 2016], and consider them as representative of the modern CNNs. They have used LeNet-5 and have simplified some of its unnecessary processes and made some improvements like normalization and anti-overfitting mechanisms: In the Alex-Net the layers are divided into two groups in which activations are calculated individually except in the first and last layer. In most layers there are no relations between the two groups in order to reduce communication traffic between the GPUs. A GPU (Graphic Processing Unit) is logic chip that is specialized in diplaying functions; it renders images, animation and video for the computer′s screen.
The overall network has a 150528-dimensional input layer, and the number of neurons in the remaining layers is given by 253440, 186624, 64896, 43264, 4096 and 1000 re-

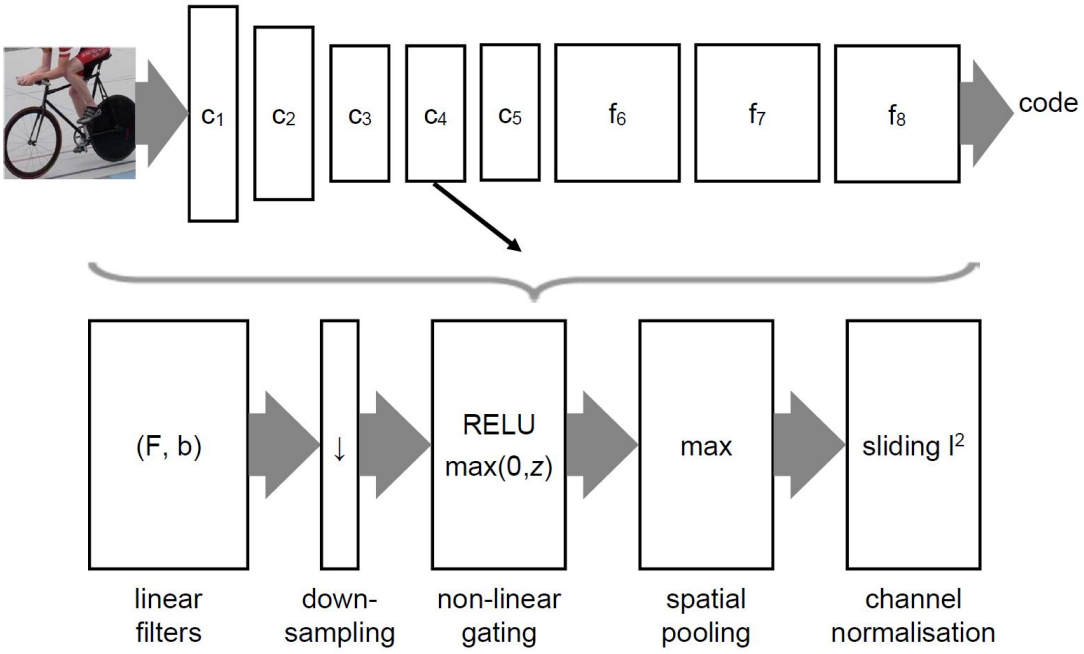


Figure 3.4: Example of CNN [Vedaldi, 2015]

spectively. The way of diving layer into groups, the number of neurons in each layers and the layer type are all alternative, thus different architecture could be devised for different problems and for different GPUs. They ask for a compromise between reducing the time cost and performance of each epoch, to make the network converges faster. To sum up, CNN have four characteristics: 1-Using convolution/local receptive field to share weights. 2-Using sub-sampling/pooling. 3-Having 2-5 convolutional layers and 1-2 fully connected layers followed by 4-Learning weight parameters by hierarchical first order optimization algorithms such as BP algorithm.

3.3 Feature Extractor

A feature extractor and a classifier are two essential modules in image pattern recognition system. An extraordinary feature extractor could produce a feature representation with more discriminant information and less correlations than the original pixel data so that the job of classifier becomes simpler and more efficient and a super classifier could perform its job well using no complex feature extractors. We know few popular techniques of feature extractor such as SIFT and HOG. Nowadays some novel classifiers and feature

extractors based on deep learning may show us fantastic results. However, if we want to create a reliabe extractor, it could be easily got by cutting the Soft-Max layer off at the end of CNNs and keeping the rest layer′s trainable parameters fixed. The features from one of each hidden layers, especially from the last one, could be invoked as the inputs of other. It means that when we train a classifier we can achieve a feature extractor inside the classifier at the same time. Based on it Zeiler [Zeiler and R., 2011] proposed the theory of feature generalization. It means that abundant feature information is included in nature images which also have a large scale of categories. Then a pre-trained network for natural images could be applied to the processing of specific data conveniently. It is important to say that the method above mentioned applies GPU based high performance computing techniques and it accelerates the training speed. In the case of the facial expression analysis the most well-know studies (the ones by Ekman [Friesen and Ekman, 1976] and Sun and Jin [Sun and Jin, 2016]), in their studies feature extractor and classifier are both sensitive to the position and shape of the five senses. In the case of facial expression classification, feature extractor and classifier should keep invariant between different individuals, and should keep invariant in the conditions of different perspective projection distortions.

3.4 General Description of MatConvNet

There are many toolboxes in Matlab to implement different kinds of Neural Networks, but as mentioned above, we are going to work mainly with Convolutional Neural Networks, so we are going to use that implementation in Matlab, which is called MatConvNet. [1].

MatConvNet is an implementation of Convolutional Neural Networks for MATLAB, whith an emphasis on simplicity and flexibility. It is composed by multiple simple building blocks that are easy to use functions such as RELU function, feature pooling etc., that are expressed as MATLAB commands, and they can be combined to create the CNN architectures.

3.4.1 Building Blocks

This toolbox is componed by building blocks which are simple really efficient functions like for example ReLUs (Rectifier Linear Units who compute the f(x)=max(0,x) function) which can be combined to create more complex algorithms. Some of the most

[1]The version of the MatConvNet used to develop the examples is the 1.0-beta20

popular ones are vl_nnconv (convolution), $vl_nnconvt$ (convolution transpose or deconvolution), vl_nnpool (max and average pooling), $vl_nnsigmoid$ (sigmoid activation) or $vl_nnbnorm$ (batch normalization). This blocks are usually written in MATLAB with the structure y=vl_nn<$block$>(x,w), but can be written in C++ and in CUDA, because MATLAB has support for GPU computation, so that it is possible to write new blocks while keeping the computational efficiency. Each of the blocks is able to function also in the backward direction to compute the derivatives. This can be done by passing a third optional argument $dzdy$, which represents the output of the network with respect to y (the input data and parameters).

3.4.2 Wrappers

A wrapper is a function or script that calls another function. This is useful because, for example, if we have different wrappers, they can call the same simulation with different parameters, and keeping different wrappers is a way of keeping track of what simulations you have. *varargin* and *varargout* are an option to write wrapper functions that accept up to 64 inputs and pass them directly to another function. In MatConvNet there are two kinds of wrappers: SimpleNN, which is the usual one in the computational blocks, and DagNN which is a more complex wrapper that accepts arbitrary graph topologies. An example of a really used SimpleNN wrapper is $vl_simplenn$ which takes as an input a net, input x and potentially outputs $dzdy$ depending on the implementation.

3.5 PreTrained Models

There are in the webpage of MatConNet [Vedaldi and Lenc, 2015] with pretrained models to classify images or to produce image encondings. However it is possible that the images to work with any of this pretrained network are needed to be This pretrained models are focused on a specific kind of photos, like for example the one that detects whether an image contains letters or no, or one to extract face features. There is also a huge image classification comptetition called ImageNet ILSVRC, and there are also some neural networks from there like for example the GoogLeNet or AlexNet. However, in our experiments, since we are going to focus on faces, we are going to use th VGG-Face, which is trained using images of the Internet Movie Data Base celebrity list.

4. CHAPTER

Facial Expression Recognition

Pantic and Bartlett [Pantic and Barlett, 2007] explain in the book Face Recognition the current state of the question of machine analysis of facial expressions. It is well known that the human face is used for human beings to gather information of other members of our species and that it is the most important means of communicating an understanding affective estates and intentions.

Our face is a communicative system capable of huge flexibility and specificity, and conveys information via theses four kind of signals:

- **Static facial signals:** Permanent features of the face.
- **Slow facial signals:** Changes in the appearance of the face that occur gradually over time.
- **Artifcial signals:** Exogenous features of the face.
- **Rapid facial signals:** Visually detectable temporal changes in the facial appearance.

If we combine this information with the Technology, and especially with Computer Science, we can see that facial expressions provide a way to communicate basic information about needs and demands to machines. For example if we combine facial spotting with facial expression interpretation it could be used for monitoring human reactions during videoconferences. Similarly, automated detectors of fatigue, depression and anxiety could

form another step toward personal wellness technologies. It is obvious that cognitive and medical scientists are interested on it and it is interesting too for lawyers, police, safe industrial systems for workers, vial security systems which check for fatigue in drivers etc.

In fact, there is a relatively recently initiated research area of affective computing that lies on sensing, detecting and interpreting human affective states and devising appropriate means for handling this affective information in order to enhance current Human Computer Interaction designs (HCI). However, although humans analyze face expressions with no effort, the development of an automated system that accomplishes this task is really difficult.

Following with the research of Pantic and Stewart, recent advances in machine analysis of facial expressions focus on four areas: Face detection, Facial feature extraction, Facial muscle action detection and Emotion recognition.

4.1 Action Units and Emotions

There are two main streams in the automatic analysis of facial expressions: one works to detect facial affect (emotion) and involves facial affect recognition methods. The other works on techniques for facial muscle action detection (action unit, AU). They come from two different approaches to facial expression in psychological research: one is all about interpretation, and says that the aim is to infer what underlies a displayed facial expression; it is called ′Message judgment′. The other attempt to be objective and his aim is to describe the surface of the facial movement o facial component shape and is called ′Sign judgment′.

Most facial expressions analyzers differentiate six basic emotions: Fear, Sadness, Happiness, Anger. Disgust and Suprise. Automatic detection of these six basic emotions in controlled displays can be done with accuracy, but in environments of real applications is a more difficult problem, and it is beginning to be explored.

In Sign judgement approaches, there is a widely used method for manual labeling of facial actions, and it is the FACS (Facial Action Coding System). It defines 44 different action units (AUs, which are the smallest visually discernable facial movements) and sets rules for recognition of AU's temporal segments (onset, apex and offset) in a face video. It provides an objective and comprehensive language for facial expressions descriptions, and it allows discovery of new patterns related to emotional states.

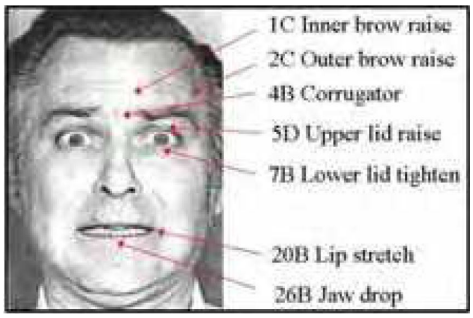


Figure 4.1: Examples of FACs code for fear)

Subjective judgement of expressions is less reliable than objective coding for finding relationships between facial expression and other state variables.

Three research groups are the forerunners in Sign judgement: Bartlett et al. [Bartlett et al., 1996], Lien et al. [Lien et al., 1998], and Pantic et al. [Pantic et al., 1998]. They use different strategies including expert rules and machine learning methods as neural networks, and they use feature-based image representations like facial points or appearance-based image representations, including wrinkles and furrows of the face.

Figure 4.2: Examples of facial action units (AUs)

The main criticisms that these research works received from cognitive and computer scientists is that they are not applicable in real-life situations, when facial expressions change rapidly. On the other hand, several works have recently emerged on machine analysis of AUs in spontaneous facial expression data (Cohn et al. [Cohn et al., 2004a], Bartlett et al. [Bartlett et al., 2005], and Valstar et al. [Valstar et al., 2006]). These works employ probabilistic, statistical and ensemble learning techniques, which seem to be suitable for automatic AU recognition from face image sequences.

4.2 Posed Facial Displays vs. Spontaneous Facial Displays

It is important to make a distinction between spontaneous facial behavior or deliberately displayed facial behavior. On one hand the volitional facial movements originate in the cortical motor strip and they tend to be less smooth, with more variable dynamics; on the other hand spontaneous facial movements originate in the subcortical areas of the brain and they are synchronized, smooth, symmetrical and consistent.

Furthermore, there is another question about facial behavior: it is biologically driven or socially learned? Researchers agree that most types of facial expressions are learned, like language, and have culturally specific meanings linked to their cultural context. On the other hand, there are a limited number of facial expressions of emotion that appear to be biologically produced, and similar across all cultures, for example anger, contempt, disgust, fear, happiness, sadness and surprise.

There is one only reported method to automatically discern spontaneous from deliberately displayed facial behavior and is the one of Valstar et al [Valstar and Pantic, 2006]. It employs parameters like speed, intensity, duration and the occurrence of brow actions to classify them as deliberate or spontaneous facial actions. For example, it has been shown that the differences between spontaneous and deliberately displayed brow actions (AU1, AU2, AU4) is in the duration and the speed of onset and offset of the actions, and in the order and the timing of actions' occurrences.

4.3 Facial Expression Configuration and Dynamics

Automatic recognition of facial expression configuration (in terms of AUs) has been the focus of the research efforts. They are important because the dynamics of facial expression are essential for categorization of psychological states like various types of pain or mood. They are also a key parameter in differentiation between posed and spontaneous facial displays (for example spontaneous smiles are smaller in amplitude, longer in duration and slower in onset and offset time than pose smiles).

Recent studies analyze explicitly the temporal dynamics of facial expressions. They explore the automatic segmentation of AU activation into temporal segments (neutral, onset, apex, offset) in frontal-and profile-view face videos. Pantic & Patras [Pantic and Patras, 2005] employ rule-based reasoning to encode AUs and their temporal segment. By the contrast,

biologically inspired learning techniques, such as neural networks, employ rule-based techniques, which emulate human unconscious problem solving processes, inspired by human conscious problem solving processes. Studies in cognitive sciences (Ambady & Rosenthal [Ambady and Rosenthal, 2005]), suggest that Learning techniques inspired by human unconscious problem solving processes may be more suitable for facial expression recognition than those inspired by human conscious problem solving. Valstar & Pantic [Valstar and Pantic, 2006] also presented evidence supporting this assumption.

4.4 Facial Expression Intensity

Researchers say that expressions can vary in intensity. Then we need to define intensity: it is the relative degree of change in facial expressions as compared to a related relaxed, neutral facial expression.

How can we measure the intensity? It is possible by using the FACS (Facial Action Coding System), which provides a 5-point intensity scale to describe AU intensity variation and enable manual quantification of AU intensity (Ekman et al. [Ekman et al., 2002]) fully automated methods that accomplish this task are yet to be developed.

4.5 Facial Expression Intentionality

How can we determine which type of message a shown facial expression communicates? To interpret it is important to know the context in which the observer signal has been displayed, where the expresser is, what his or her current task is, are other people involved and who the expresser is.

4.6 Context Dependency

Since the problem of context-sensing is extremely difficult to solve, we need pragmatic approaches when learning the grammar of human facial behavior. Unfortunately existing automated facial expression analyzers are context insensitive.

To develop facial behavior analyzer machines we need to take into account some factors

that influence affective data collection. Thinking about it, Picard [Picard, 1997] outlined five factors:

- Spontaneous versus posed emotions.
- Lab setting versus real-world.
- External expression or internal feeling.
- Does the subject know that is being recorded or not?
- Does the subject know that he is a part of an experiment about emotion?

4.7 Databases

There are some isolated pieces of facial database:

- One is the Ekman-Hager Facial Action Exemplars (Ekman et al. [Donato et al., 1997]) which has been used by several research groups to train and test their methods for AU detection from frontal-view facial expression sequences.
- We have too the JAFFE database (Lyons et al. [Lyons et al., 1999]), which contains 219 static images of 10 Japanese females displaying posed expressions of six basic emotions.
- Other one is the Cohn-Kanade facial expression database ([Kanade et al., 2000]), which is the most widely used in research on automated facial expression analysis and contains image sequences of 100 subjects posing a set of 23 facial displays, and contains FACS codes in addition to basic emotion labels.
- We have also the most comprehensive database for research on automated facial expression analysis, which is the MMI facial expression database (Pantic et al. [Pantic et al., 2005]), but it still lacks metadata about the context. It has two parts: one first part with deliberately displayed facial expressions (over 4000 videos and 6000 static images of single AU, multiple AU and six basic emotions; and a second part with Spontaneous facial displays (65 videos of spontaneous facial displays in terms of AUs and emotions, in which were 18 adults and 11 children , male and female, Caucasian, Asian and African).

- Other database is UT Dallas, similar to the second part of the MMI facial expression database, and works in terms of AUs and emotions, independently of the stimulus category is needed.
- There is other one called the RU-FACS Spontaneous Expression Dataset and consists of 100 subjects participating in 'false opinion' paradigm.
- Finally, the database used in our project is the Child Affective Facial Expression (CAFE), from the Depearment of Psychology at Rutgers University Newark in New Jersey (USA) and it will be explained in other part of the work set

Except of the MMI facial expressions database, which was built as a web-based direct-manipulation database, the existing facial expression databases are neither easy to access nor easy to search.

4.8 Face Detection and Feature Extraction

We have numerous techniques for face detection in images, but most of them can detect only upright faces in frontal or near-frontal view. Different methods are:

- Rowley et al [Rowley et al., 1998] used a multi-layer neural network to learn the face and non-face patterns from the intensities and spatial relationships of pixels in face and non-face images.
- Moghaddam and Pentland [Moghaddam and Pentland, 1997] developed a probabilistic visual learning method based on density estimation in a high-dimensional space using an eigenspace decomposition. They used it to face localization, coding and recognition.
- Scheiderman and Kanade [Schneiderman and Kanade, 2000] developed Statistical method for 3D object detection.
- Viola and Jones proposed the most commonly used method: it is a real-time face detector. It has several adapted versions, as the one of AdaBoost, which can employ image filters and permit a high speed of the detector.

When the machine has detected the presence of a face, the next step is to extract the information from it. The problem of facial feature extraction from the scene may be divided into, at least, three dimensions:

- Is temporal information used?
- Are the features holistic (spanning the whole face) or analytic (spanning subparts of the face)?
- Are the features view-or volume based (2D/3D)?

Most of the facial expression analyzers are directed toward 2D facial feature extraction. The extracted facial features are one of these:

- Geometric features, such as the shapes of the facial components (eyes, mouth, etc.), and the location of facial points (mouth, corners of the eyes, etc.)
- Appearance features, which represent the texture of the facial skin including wrinkles, bulges, and furrows.

Approaches to the facial expression analysis based on 3D face modelling have been recently proposed. Gokturk et al. [Gokturk et al., 2002], Cohn et al. [Cohn et al., 2004b], Gross et al. [Gross et al., 2006] and they are very relevant to produce view independent facial signal recognition systems, but they need a large amount of manually annotated training data and a manual selection of landmark facial points in the first frame of the input video, based on which the model will be warped to fit the face.

4.9 Geometric Facial Feature Extraction and Facial Point Detection

Previous methods are either texture-based methods (modeling local texture around a given facial point) or texture- and shape-based methods (learned from a set of labeled faces and trying to fit the shape to any unknown face). A texture based method was applied by AdaBoost to determine facial feature point candidates for each pixel in an input image and used a shape model as a filter. This method uses 20 facial characteristic, but they regard the localization of a point as a SUCCESS if the distance between the automatically labeled point and the manually labeled point is less than 30

To handle this issue Vukadinovic and Pantic [Vukadinovic and Pantic, 2005] developed a novel, robust, fully automated facial point detector. It is a textures based method -models local image patches using Gabor wavelets and builds GentleBoost-based point detectors

based on these regions. The detected face region is divided in 20 regions of interest (ROIs), each corresponding to one facial point to be detected. A combination of heuristic techniques based on the analysis of the vertical and horizontal histograms of the upper and lower half of the face region image is used for this purpose. Gabor features are among the most effective texture-based features for face processing tasks. It is so because Gabor filters remove most of the variability in image due to variation in lighting and contrast, and at the same time they are robust against small shift and deformation. In the training phase GentleBoos feature templates are learned using a representative set of positive and negative examples. In the testing phase an automatically detected point displaced in any direction, horizontal or vertical, less than 5% of inter-ocular distance from the true facial point is regarded as SUCCESS. Overall, an average recognition rate of 93% was achieved for 20 facial feature points. Fasel and colleagues developed a real-time feature detector using a GentleBoost approach related to the one used for their face detector and combined with a Bayesian model for feature positions. (Fasel, 2006). The Face is first detected and then the location and scale of the face is used to generate a prior probability distribution for each facial feature.

4.10 Appearance-based Facial Features

Humans can recognize facial expressions above chance from motion, using point-light displays. However humans are very good for recognizing expressions from texture without motion, for example static photographs.

Most computer vision researchers consider the problem of facial expression recognition and in the appearance-based features include: Gabor filters, Integral image filters (also known as box filters and Haar-filters), Features based on edge-oriented histograms and Active Appearance Models, spatio-temporal features. Active Appearance Models include Motion energy images and motion history images, Learned image filters from independent component analysis (ICA), Principal component analysis (PCA), Local feature analysis (LFA) and Linear discriminant analysis (e.g. fisherfaces).

A reservation about appearance-based features for expression recognition is that they are affected by lighting variation and individual differences. However, machine learning systems taking large sets of appearance-features as input, and trained on a large database of examples, are emerging as robust systems in computer vision.

The importance of appearance-based features for expression recognition is emphasized

by several studies that suggest that appearance-based features may contain more information about facial expression than displacements of a set of points. Bartlett and colleagues (Dionato et al. [Donato et al., 1999]) compared a number of appearance-based representations on the task of facial action recognition using a simple nearest neighbor classifier. They found that Gabor wabelets and ICA gave better performance than PCA, LFA, Fisher discriminants, and also outperformed motion flow field templates. More recent comparisons found an interaction between feature-type and classifier, where AdaBoost performs better with integral image filters, while SVMs performs better with Gabors. The possible motive is that AdaBoost performs feature selection and does well with redundancy whereas SVMs were calculated on the full set of filters and don't go well with redundancy.

4.11 Appearance-based Facial Affect Recognition

The appearance-based facial expression recognition system developed by Bartlett et al. [Bartlett et al., 2003] is a system that automatically detects frontal faces in the video stream and codes each frame with respect to 7 dimensions: neutral, anger, disgust, fear, joy, sadness, surprise. It operates in near-real-time, and first performs automatic face and eye detection using the appearance-based method of Fasel et al. [Bartlett et al., 2005], then select extracts subsets of the features and passes them to an ensemble of classifiers which make a binary decision about each of the six basic emotions plus neutral. Best results were obtained by selecting a subset of Gabor filters using AdaBoost and then training SVMs on the outputs of the filters selected by AdaBoost. The combination of AdaBoost and SVMs enhanced both speed and accuracy of the system.

4.12 Facial Muscle Action Detection

Although FACS provides a good foundation for AU-coding of face images by human observers, achieving AU recognition by a computer is a difficult task. It is so because AUs can occur in more than 7000 different complex combinations, causing bulges and movements of permanent facial features (for example jetted jaw) that are difficult to detect in 2D face images There are 2 main groups who do research in this field: Pantic and her colleagues, and Barlett and her colleages.

4.13 Feature-based Methods for Coding AUs and their Temporal Segments

Pantic and her colleagues wanted to automate the analysis of facial expressions in terms of facial muscle actions that constitute the expressions. Work was aimed at AU coding in static face images. Recent work addressed the problem of automatic AU coding in face video. They experimented with rule-based and Support Vector Machine based methods for recognition of AUs in either near frontal-view or near profile view face image sequences. The methods proposed address the problem of temporal modeling of facial expressions. These methods are suitable for encoding temporal activation patterns (onset - apex - offset) of AUs shown in an input face video. Nowadays the only systems to date for explicit recognition of temporal sements of AUs are the ones by Pantic and colleagues (2005)

It is necessary to understand how to achieve automatic AU detection from the profile view of the face to build a technological framework for automatic AU detection from multiple views of the face. Pantic and Patras (2006) proceed under two assumptions: one is that the input image sequence is non-occluded (left or right) near profile-view of the face with possible in-image-plane head rotations, and the other is that the first frame shows a neutral expression.

This method for AU coding in near profile-view face video was tested on MMI facial expression database only, and the accuracy of the method was measured with respect to the misclassification rate of each ′expressive′ segment of the input sequence. For 96 test samples they achieved an average recognition rate of 87% for 27 different AUs. occurring alone or in combination in an input video.

Another system created by Valstar and Pantic used a System that detects AUs and their temporal segments (neutral, onset, apex, offset) using a combination of Gentle Boost learning and Support Vector Machines (SVM). GentleBoost is used to select the most informative features. An AU can be in four different phases 1-Onset phase: Muscles are contracting and the appearance of the face changes as the facial action grows stronger; 2-Apex phase: The facial action is at its apex and there are no more changes in facial appearance due to this particular facial action; 3-Offset phase: the muscles are relaxing and the face returns to its neutral appearance; 4-Neutral phase: There are no signs of activations of the facial actions;

As every facial action can be divided into four temporal segments (neutral, onset, apex,

offset) Valstar and Pantic consider the problem to be a four-valued multi-class classification problem and they use a one-versus-one approach to multi-class SVMs (mc-SVMs).Experiments were done on MMI database only and there was a 95% precision.

It seems that human observers detect activation of these AUs based on the presence of a certain movement like an upward movement of the lower eyelid, but also based on the appearance of the facial region around the eye corner, like the crow feet wrinkles. Such an appearance change may be of a different duration from the movement of the eyelid resulting in an erroneous estimation of AU duration by the system that takes only facial movements into account. Because of that using geometric and appearance features might be the best choice in the case of such AUs

4.14 Appearance-based Methods for AU Coding

Barlett and colleagues developed an appearance-based system for fully automated facial action coding developed. It is user independent and operates in near-real time, at about 6 frames per second. The system detects 30 AUs. The system captures information about AU intensity, that can be employed for analyzing facial expression dynamics. Pantic used heuristic, rule based methods, and/or designing special purpose detectors for this methods. Barlett uses machine learning. Over 200 examples are needed to obtain high precision.

Kappor (2003) created another appearance-based system for fully automated AU coding. The system uses infrared eye tracking to register face images employing machine learning techniques on feature-based representations. In the research of Pantic and Bartlett it was trained with 2568 examples from 119 subjects. Positive examples consisted of the last frame of each sequence which contained the expression apex. Negative examples consisted of all apex frames that did not contain target AU plus neutral images obtained from the first frame of each sequence, for a total of 2568-N negative examples of each AU.

The system obtained a mean of 91% agreement with human FACS labels. In the test, there was far grater number of non-targets that were all images not containing the desired AU (2568-N). A more reliable performance measure is area under the ROC (receiver-operator characteristic curve, or A'). This curve is obtained by plotting hit rate (true positives) against false alarm rate (false positive) as the decision threshold varies.

Correlations of the automated system with the human expert intensity scores were next computed: Mean correlation between the SVM margin and the expert FACS coders was 0.83 which is nearly as high as the human-human correlation of 0.84. Similar findings

were obtained using an AdaBoost classifer, where the AdaBoost output, which is the likelihood ratio of target/non target correlated positively with human FACS intensity scores.

Test were also done in the RU-FACS Dataset of spontaneous expressions. Mean area under the ROC for the spontaneous action units was 0.75. The SVM margin correlated positively with AU intensity in the spontaneous data.

4.15 Automatic Detection of Pain

The automated AU recognition system described above was used to differentiate faked from real pain expressions using the automated AU detector. Each subject experienced three experimental conditions: baseline, real pain, and posed pain. The trained linear SVM for each of 20 AUs in one versus all mode, irrespective of combinations with other AUs. The results were passed to another set of three SVMs, trained to detect real pain, fake pain and baseline. In a preliminary analysis of 5 subjects tested the system correctly identified the experimental condition for 93% of samples in a 3-way forced choice. The 2-way performance for fake versus real pain was 90. This is considerably higher than the performance of naive human observers.

4.16 Challenges, Opportunities and Recommendations

Pantic and Bartlett [Pantic and Barlett, 2007] summarize the recent work of two forerunning research groups in this research field, namely that of Pantic and her colleagues and that of Barlett and her colleges

The research on automatic detection of facial muscle actions, which produce facial expressions like happiness and anger, witnessed a significant progress in the past years.

The majority of the past work in the field does not analyze the properties of facial expression temporal dynamics, but there are a few approaches to automatic segmentation of AU activation into temporal segments (neutral, onset, apex offset)

Also, even though most of the past work on automatic facial expression analysis is centered on posed facial expressions, there are a few efforts on machine analysis of spontaneous facial expressions. Similarly there are a few works on context-sensitive interpretation of facial expressions and an attempt to discern in an automatic way spontaneous from volitionally displayed facial behavior.

When it comes to automatic AU detection, existing methods do not yet recognize the full range of facial behavior. The use of a combination of methods based on geometric features and methods based on appearance features is necessary if the full range of human facial behavior is to be coded in an automatic way.

Existing methods for machine analysis of facial expressions discussed throughout the research of Pantic and Bartlett assume that the input data are near frontal- or profile-view face image sequences showing facial displays that always begin with a neutral state. In reality, such assumption cannot be made. Noisy and partial data should be expected.

Probabilistic graphical models, like Hidden Markov Models (HMM) and Dynamic Bayesian Networks (DBN) are well suited for accomplishing a statistical prediction and its probability by considering previously observed data (time scale) with respect to the current data (time instance). These models can handle noisy features, temporal information and partial data by probabilistic inference.

It remains unresolved how the grammar of facial behavior can be learned (in human-centered manner or in an activity-centered manner) and how this transformation can be properly represented and used to handle ambiguities in the observation data. Another related issue that should be addressed is how to include information about the context: environment, user, user′s task, so that a context-sensitive analysis of facial behavior could be achieved.

They believe that a large, focused interdisciplinary, international program directed towards computer understanding of human behavioral patterns, as shown by means of facial expressions, should be established if we want to achieve breakthroughs in this field.

5. CHAPTER

DB for the project

5.1 Motivation

Most of the previous work done in face analysis it is done with faces of adult people, however, due to the problems of doing research with underage people among other things, there are not many databases with faces of children.

5.2 Paperwork

Due to the problems of having photos of underage people, it was neccesary to ask for a written permission to the New York University, where it is stated that the images of the database are going to be used only for this research and no other purposes, and it is stated that it is forbidden to show any image of the children in any report or article. After that a permission was granted to access the database through Databrary, which is a video data library for developmental science.

5.3 Structure and content

The database used in this project is The Child Affective Facial Expression (CAFE) [LoBue, 2014] set which is a collection of photographs of 2- to 8-year old children with

a median M=5.3 years and with a rho r=2.7-8.7 years. This dataset of the Department of Psychology at Rutgers University Newark in New Jersey contains posing for 6 emotional facial expressions according to Ekman (PUT REFERENCE): sadness, happiness, surprise, anger, disgust and fear plus a neutral face. The set is composed of photos of 154 models: 90 female models and 64 male models (27 African American, 16 Asian, 77 Caucasian/European American, 23 Latino and 11 South Asian). The structure of the database in the downloaded zip is the following:

- materials-9822-Top-level_materials
  - 11185 – $Sortable\_Excel\_file\_with\_basic$
  - 13855 – $LoBlue\_Thrasher, \_2015$
- sessions
  - 6280
  - 6281
  - ...
  - 6435
  - 6436
- description.html
- spreadsheet.csv

Each of the folders represents a set of photos of a children, but all of the folders does not contain the same number of photos, because there are not photos of each children with all the emotions. There are two kinds of photos of emotions, one is with the mouth open and the other one is with the mouth closed. Each of the folders can have photos of the subject expresing emotions with the mouth open or closed indiferently, but there can be cases where all the photos of the subject are with the mouth open or with the mouse close, and there might not be all the emotions in a subject. All of the photos included in the dataset contain the image centered in the child's face, with their chin approximately 1/6 from the bottom of the image, and all of the images contained FACS codes because all the photos that did not include any were deleted from the original database. In order to make the images have this codes, the photographer made the children copy the faces and tell them which emotions to show in the face, and if any elements were missing, the photographer

prompted the children to revise their facial expressions [Lobue and Thrasher, 2015]. This photographer was trained in SPAFF which is a system that includes procedures for recognizing facial muscle movements associated with 17 codable emotional states in real time, and also incorporates the FACS coding system of Ekman and Colleges [Ekman, 1992]. This database originally is composed by two subsets: Subset A which contains the stereotypical exemplars of the various facial expressions, the "basic" emotions we explained previously, and Subset B expressions that vary around the "basic" expression, but minimizing potential ceiling and floor effects. However, this division is not used in our experiments.

5.4 Connection of the Database with Matlab

In order to make it easier to work with the database it is a good idea to connect Matlab to MySQL. To do that there are two options: ODBC or JDBC. ODBC stands for Open Database Connnectivity and it is a standard application programming interface for accesing database management systems (DBMS). ODBC was developed by SQL Access Group in 1992 when there were no ways of communicating between a database and an application. It is not dependent of a specific programming language or a database system or operating system, but since Microsoft adapted to it, it is mostly used in Windows. It is procedural. Java Database Connectivity is an application programming interface for Java programming language. It is suitable for object oriented databases, and there is a JDBC-to-ODBC bridge that enables connections to any ODBC-accesible data source.

ODBC is slower than JDBC, but since in this case the connection was going to be made with Matlab and it uses some C++ features, ODBC was chosen. In this case the development was made in a Windows platform, however there are also Linux ODBC drivers.

In order to connect the database from Matlab, first of all it is necessary to have a database schema created in MySQL (Check image 5.1).

After having that, go to the App tab and click in the Database Explorer (Check image 5.2).

Cancel the emerging window and after that click in the Database Explorer tab, and click on New->ODBC to create a new ODBC connection from Matlab (Check image 5.3).

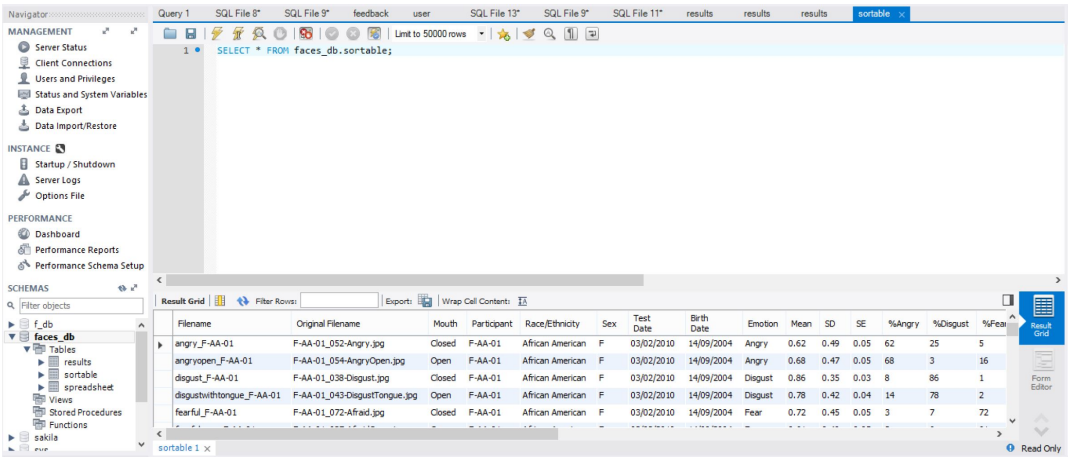


Figure 5.1: MySQL Database

Figure 5.2: App->DatabaseExplorer

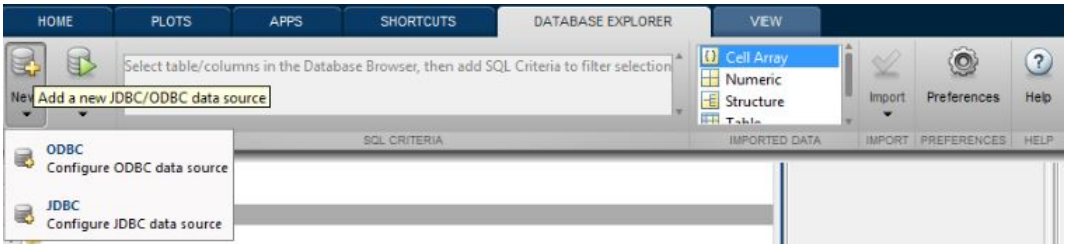


Figure 5.3: Database Explorer menu

After that go to the DNS System tab, and if you see your MySQL Schema name, that means that your that schema is already linked to the Matlab environment (Check image 5.4).

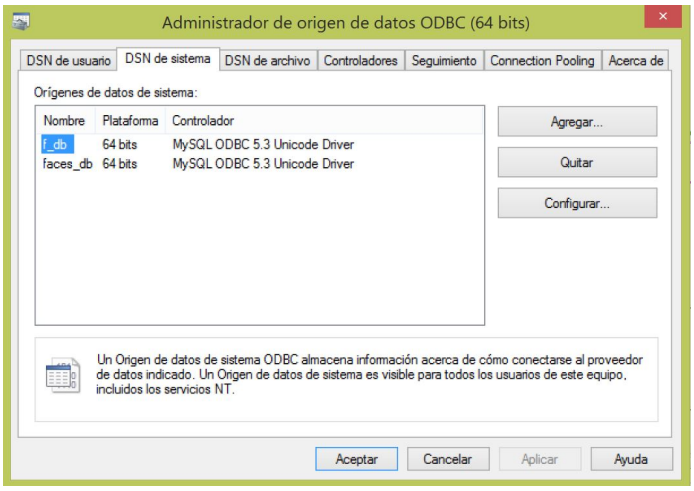


Figure 5.4: System DNS

If your MySQL Schema is not in there, you should click the add button and fill the gaps in the next window. (Check image 5.5).

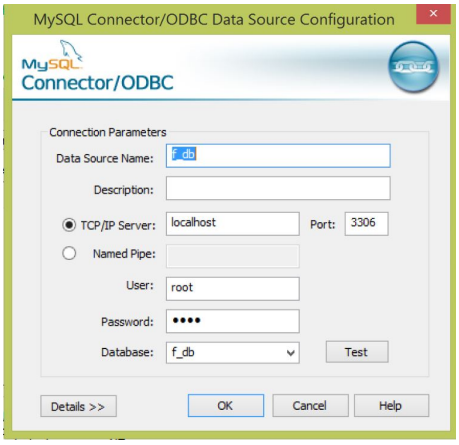


Figure 5.5: Database Explorer menu

After that you will be able to see the connected database in Matlab, like in the next image (Check image 5.6).

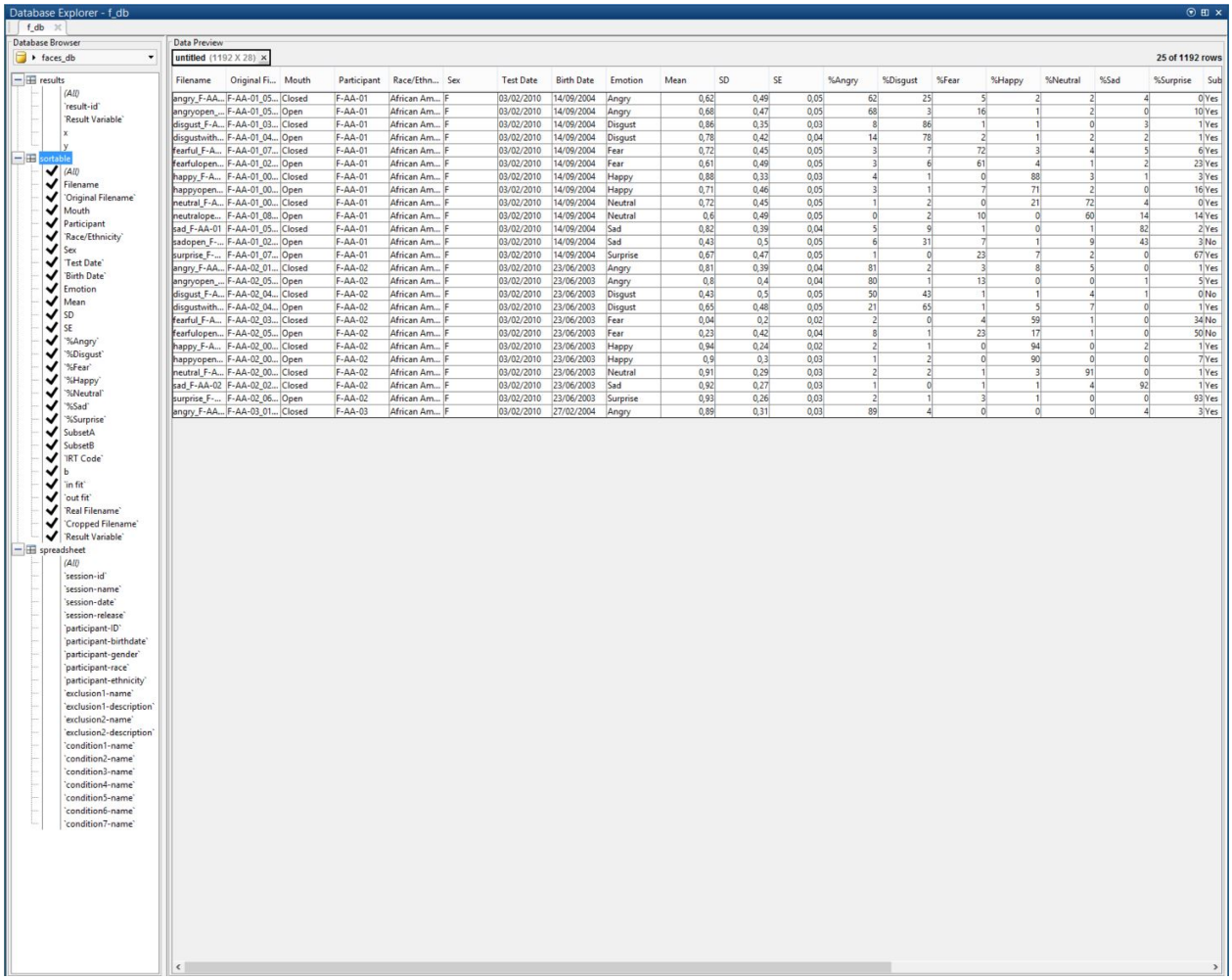


Figure 5.6: Database in Matlab

Then in order to manage the queries it is necessary to make a connection to the database with the function *database*, create a pointer with the query and bring all the information that is needed. It is possible also to add, delete or update information from this tables by creating the corresponding query, or by creating the changes locally in the tables that we charge in Matlab and then using *update* with the database connection opened to upload the changes to MySQL.

```
conn = database(db_name, 'root', 'root');
if (isempty(conn.Message))
    disp('Database connected);
else
    disp('Cannot connect database');
    disp(conn.Message);
    return; % stop running
end

sql = sprintf('SELECT `session-id`, `participant-ID` FROM
   spreadshhet');
curs = exec(conn, sql);
curs = fetch(curs);
sprdata = get(curs, 'Data');
```

5.5 Check DB and prepare it for experiments

In order to check that everything was alright on the database and prepare it for the experiments, some Matlab scripts were created. This scripts made calls to the MySQLWorkbench, and do modifications in the tables in order to make it easier to work with the database or to check everything is alright.

The first script checkDatabase.m takes care of checking that every photo has a assigned a user that exists. It creates a new column to check depending on if it has or not a user that exists writes "Yes" or "No" in the column and then counts the number of "Yes" to check wether is the same as the number of users, to check at the same time whether all the users are used (After execution this column is deleted). Doing this, a problem appeared:

In the table sortable, there were some participant with a participant identifier ending with the letter 'b', however, in the table spreadsheet, there were no participant identifiers that match those ones. However, there were some of the identifiers repeated (some of those ones are the ones that should have the 'b' at the end). So, I added the 'b' at the end of the identifiers in the spreadsheet table, after this change everything is properly configured, and that there are no users without images or images have not a user asigned in the other table.

The second script checkAllImages.m gives each table the real filename, because the one in the table is not the original one.

The third script cropImages.m creates a new folder called 'sessions_cropped' that includes all the original images cropped and centered into the children faces. It also takes care of reducing the scale of the too big images. This process of centering in the face of the children and cropping it sometimes fails, because the bounding box focuses too much in a certain place confusing it with the image of the whole face.

In order to detect this outliers and delete them a query was used to detect the files to delete

```
1 SELECT COUNT(DISTINCT `Filename`) AS r FROM faces_db.sortable s INNER JOIN (SELECT `result-id`
      FROM faces_db.results WHERE faces_db.results.x >650 AND faces_db.results.y>650) z ON s.`
      Filename`=z.`result-id`;
```

and these lines were deleted also from MySQL. This script checks the size of the images, because when cropping the images if the bounding box is centered in a really specific area it is going to cut that small area. So, those images whose size is considerably smaller than the rest are the images that should not be taking into consideration (Check Figure 5.1).

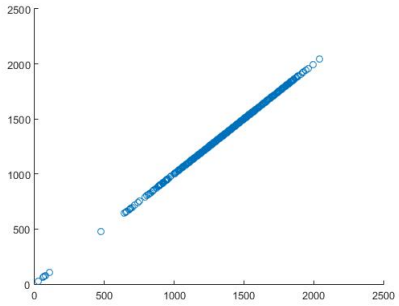


Figure 5.7: Image outliers

As seen in the figure, the images with a size less than 500x500 pixels, should be discarded. However, there is one image, that should be checked manually, because it is in the limit.

After checking it, it turned out that the image is also not valid, because although a part of the face appears, the eyes and the mouth are cutted. So, the images that should be taken are the ones with more than 650x650 pixels.

Finally, in order to save the features of each of the images another script was create saveFeatures.m. This script uses a pre-trained Convolutional Neural Network in faces into the images of the database [Parkhi et al., 2015], and extracts the features of each one of this images. This results are stored inside the folder of cropped images created before, and the names of the variables are stored in the tables to manage the DBs.

6. CHAPTER

Experiments

6.1 Introduction

The objective of this experiments is to check whether well does a computer recognize the emotions on childrens in the range of age 2-8, since most of the work that has been previously done it has been done on adults or children from 10 years old and up. In order to do any experiment there is always a fixed process to follow:

6.2 Preprocess of Data

First of all the data should be preprocessed, in our case as explained in Chapter 5 we cropp the images to have less background in the image and center them more in the face, and we delete the outliers that appear because of this process. In our case, we are going to do 2 experiments, one with the full dataset, and another one with one dataset containing all the images of the children with the mouth open, and another one containing all the images of the children with the mouth closed. So, this two datasets were also created.

6.3 Experimentation

The experiments were all realized in Weka with the .csv files obtained from Matlab. Different classifieres were tried both in the full dataset and in the dataset divided in two, and in the next section the most relevant ones were used to make the comparation with the rest of them and with the one in the original article that was used with the DataBase.

However, in the original article, to detect the emotions and make the statistics, 100 untrained adults were used, and they were showed the full set of images twice. These adults were undergraduate students (half male and half female and the M=21.2 years) from the Rutgers University-Newark campus. These 100 participants were 17% African American, 27% Asian, 30% White and 17% Latino (the other 9% did not indicate their race or ethnicity).

6.4 Results and Analysis

6.4.1 Full dataset

The results by humanas and by classification in the full dataset were the following ones:

Mean correct T1	**Mean correct T2**	**Std. deviation T1**	**Std. error of mean T1**	**Cronbach′s alpha (T1, T2)**
0.66	0.66	0.47	0.001	0.77

Table 6.2: Human classification general results

Name of classifier	**Correctly classified**	**Mean abs. error**	**Outliers in**
J48 10-fold cross validation	39.0428%	0.1778	Yes
SVM (Lin. Kernel) 10-fold cross val.	63.1402%	0.1053	Yes
SVM 10-fold cross val. Sigm. Kernel	63.6824%	0.10308	No
SVM 10-fold cross val. Lineal Kernel	55.1899%	0.128	No
K-Means with Crossvalidation	39.417%	0.2108	No
J48 Crossvalidation	38.0912%	0.18	No

Table 6.3: Machine Classification general results

Emotion Precission	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
J48 10-fold cross validation	0.446	0.385	0.285	0.419	0.543	0.211	0.2
SVM (Linear Kernel) 10-fold cross val.	0.562	0.584	0.576	0.73	0.729	0.571	0.467
SVM 10-fold cross validation Sigm. Kernel	0.593	0.597	0.59	0.73	0.708	0.569	0.456
SVM 10-fold cross validation Linear Kernel	0.6	0.416	0.611	0.662	0.592	0.611	0.471
K-Means with Crossvalidation	0.456	0.468	0.346	0.398	0.376	0	0.533
J48 Crossvalidation	0.411	0.33	0.233	0.507	0.507	0.198	0.299
Humans Time 1	0.66	0.64	0.42	0.85	0.66	0.62	0.72
Humans Time 2	0.65	0.66	0.49	0.83	0.65	0.63	0.65

Table 6.5: Each emotion results humans and machine classification

As seen in the 6.2 Table, the maximum mean value obtained with human classification is 66% whereas with machine classification the maximums are between 55% and 63.68%, being the best results the ones classified with SVM in combination with another classifier. It is surprising to check that even though there are some outliers that should be giving problems the SVM (Linear Kernel) 10-fold cross val, obtains the 2nd best possition in correctly classifing all the images but since the outliers are were only 8 photos, it is possible, that this fact is not affecting much. It is also surprising to check that even though the SVM 10-fold cross validation Sigmoid Kernel obtains the most corretly classified instances, it is also the one with the lowest mean absolute error. Regarding the emotions, in the human in the first tryout the classification of the emotions was slightly better than in the second one, but there were no substantial changes. It was not suprising to see that highest values of most of the emotions were in the classifications done with SVM. Nevertheless, it was not expected to see that the values obtained with the SVMs are more normalized than the ones obtained with the human classification, furthermore, it was also remarkable that the fear emotion was really well classified by the machines, but the humans failed to classified

it so accurately, but in the case of the suprise it is the other way around. This is probably becasue Surprise and Fear are two emotions that have FACS points in common and are easy to mistake them.

6.4.2 Open and close mouth dataset

The experiment with two datasets is to check whether it is easier for the to detect the emotions in pictures with the mouth opened, with the mouth closed or if there is not a big difference at all. The results obtained in the open mouth dataset were the following ones:

Name of classifier	Correctly classified	Mean abs. error	Outliers in
SVM Linear kernel. Crossvalidation	61.125%	0.1111	No
J48 (trees) Crossvalidation	34.75%	0.1989	No
KNN 7 classees Crossvalidation	31.875%	0.2168	No

Table 6.6: Machine Classification Subset mouth opened general results

Emotion Precission Open Mouth	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
SVM Linear kernel. Crossvalidation	0.563	0.53	0.556	0.705	0.663	0.29	0.802
J48 (trees) Crossvalidations	0.258	0.406	0.212	0.292	0.365	0.111	0.527
KNN 7 classes Crossvalidation	0.321	0.438	0.233	0.347	0.278	0.043	0.769
Human Time 1	0.66	0.73	0.38	0.74	0.4	0.45	0.72
Human Time 2	0.68	0.77	0.46	0.73	0.4	0.47	0.65

Table 6.8: Subset mouth opened each emotion results classified by humans and machine

The results with the open mouth dataset were worse that both the ones obtained with the humans and the best ones obtained with SVM in the full database. However, the best case in this subset was still obtained with SVM combined with another classifier. Nevertheless in this case specially the sad emotion is really low compared to the one obtained by the human classification or the ones in the full daatabase with SVM.

The results obtained by human in the closed mouth images dataset:

Name of classifier	Correctly classified	Mean abs. error	Outliers in
KNN Crossvalidation 5 classes	31.5104%	0.2992	No
SVL Lineal Kernel. Crossvalidation	59.375%	0.1625	No
J48 (trees) Crossvalidation	38.8021%	0.2572	

Table 6.9: Subset mouth closed general results in machine classification

Emotion Precission Open Mouth	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
KNN Crossvalidation 5 classes	0.614	-	0.17	0.307	0.305	0	-
SVL Lineal Kernel. Crossvalidation	0.733	-	0.563	0.496	0.639	0.45	-
J48 (trees) Crossvalidation	0.549	-	0.349	0.387	0.393	0.108	-
Human Time 1	0.66	0.54	0.45	0.93	0.86	0.75	-
Human Time 2	0.64	0.56	0.38	0.91	0.84	0.74	-

Table 6.11: Subset mouth closed each emotion results in human and machine classification

In the case of the subset with the mouth closed, the results are also worse than in the full dataset results, but they are even worse than the results in the mouth open images database, and it happens the same as before, the results with the SVM combined with another classifier are the ones having the better results.

6.5 Conclusions

Taking into consideration that the results that we are comparing with are results not created by a machine, they are created by humans, which tend to recognize emotions so precisely, the results are pretty good, because the results with the SVN Linear Kernel with crossvalidation (63% of precission), which are the best results we could get, are really close to the results of the people (66%). Cropping the image and getting it closer to the camera improved in a way the results we had at first which were around 40%. However, it must be mentioned that due to the newness of the database, that there is not further research in people so young and that the fact of extracting emotions is a complex problem,

the results should be compared in further investigations. It was also interesting to see that the used pretrained network was trained with a database of faces of older people, and even thought the results were not that bad, and it is important to notice that as people get older, the facial features modify legerely over time, whereas the face of a children has nearly no expressions in the skin. It would be also interesting if instead of using a pretrained model like the one used in the experiments in here, further investigations train a neural network with images of faces of children, to see if the results can even pass the ones from the humans. Moreover, not using only faces of children with the head properly centered, but having also photos of children with the head a bit tilted to one side would be also interesting.

Appendixes

A. APPENDIX

Planification

A.1 Description of the concrete objectives of the project

A.1.1 Characterization of the product to develop

The products to be developed are a study of the basic resources for the analysis and classification of child expressions and a fully functional program that after being trainned with some photos of children, when new photos of children are shown it can classify in which percentage this children are happy, surprised, afraid, disgusted, angry and sad,.

A.1.2 Characterization of the environment in which will be distributed

The documentation with the results of the fully developed product and the analysis of the basic resources for the analysis and classification of child expressions and the developed software itself will be uploaded in an electronic format to ADDI. And after being presented to the thesis comitte and being accepted by it, the full product will be published in the webpage of ADDI.

A.1.3 License of the product

All the documentation and the code created for this activity will be distributed under the Creative Commons Attribution license: Licensees may copy, distribute and perform the work and make derivative work based on it only if they give the author or licensor the credits in the manner specified by these. If any of this material is used (either in printed or electronic format), the authors must be acknowledged this way:

- The full names of the original authors must be recognized.
- A link to where either the documentation, the code of both were downloaded (ADDI) must be mentioned.

The attributtion and acknowledment for the development of this thesis will be mentioned in the Bibliography section.

A.2 Identification of the deliverables and its characteristics

A.2.1 Related to the object of the project itself

- Fully functional expression recognition software
- Study of the basic resources sfor the analisis of emotions from the expressions in children faces
- Slides for presenting the project in front of the thesis comitte
- Report of the project

A.2.2 Related to the management of the project

- Planification document which should contain the following:
  - Scope of the project
  - The deliverables and their characteristics
  - WBS (Work Breakdown Structure of the project

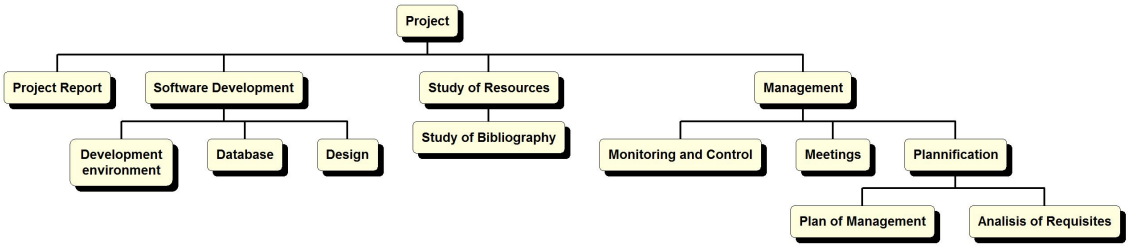
  - Timetable of the project
  - Plan of quality
  - Adquisitions

  This document should be available for the 18th of february at 15:00

- Document of monitoring and control of the project. It should contain the activities done, when they have been done and the effort (time) spend in each of them. It should also contain:

  - Significant deviations between the expected time to spend and the real time spend
  - Planification, execution, management and control of the adquisitions

A.3 WBS (Work Breakdown Structure



A.4 Quality

A.4.1 Minimum Quality

- The software is fully working
- The sofware recognizes the percentage of each of the 6 emotions in the given photos
- The software classifies the images depending on the percentages of the emotions

- The analysis contains the basic resources for the analysis of emotions from the expressions on children faces

A.4.2 Process to secure the quality

All the deliverables for this project; the ones related to the management as well as the working software and the analysis must be finnished a week before the limit of the delivery day. That gives time the director of the thesis to have a revision and a general analysis with a margin of time, in case any modification is needed. Also the weekly meetings with the director make it possible to have a control of how the project is going, in case some big deviation or problem is happening.

A.5 Study of analysis of the different databases and developing environment to be used

After considering some alternatives of Databases, the chosen database is CAFE. This decision was based on this database being free and it included photos of children frontally. Some other databases like Face Databases were found, but they contained images of adults, not children, or they did not contain faces of emotions, so they were not appropiate for this project. All the changes necessary for the analysis of the DB will be done in Matlab, and this will be connected to MySQL to make the operations easier. All the experiments will be done in WEKA.

A.6 Description of the tasks to be done

Planification:

- Evaluation of the resources, concretion of the scope, analysis of risks, minimum quality...
- Analisis of the created software
- Management of the obtaining of the database
- Write the planification

Software development:

- Study the best development environment
- Adquisition of the database
- Check the requirements of the adquired database

Monitoring and control:

- Monitoring and control of the tasks

Study of basic resources for expression analysis:

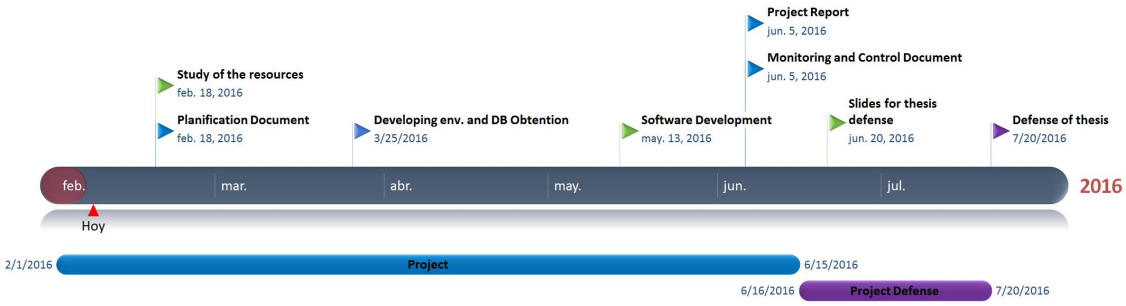
- Analysis of bibliography

Report of the project:

- Explanation of the project itself
- Theory about the project and how it is applied

A.7 Milestones diagram

First prevision



Second prevision Because of different problems happened during the month of may and june, such as problems with the use of the DB and with the experiments we decided to change the first prevision and to make a second one in which we would present the project on the month of September. As a consecuence the software development was made during the months of june, july and beginning of august; the project report and the monitoring control document were made for the beginning of september, and the slides were made for the beginning of september also.

A.8 Management of changes

At any point in the project it is possible that some kind of change it it is needed, so it is necessary to have a good reaction plan, as it is necessary to do it with the risks.

- There will be a margin every week in case more hours than the planned ones are needed

A.9 Identification of risks

- The legal difficulties for obtaining images of children
- The possibility of having a lot of knowledge in the chosen development environment
- The possibility of not finding pre-developed software, plugin or similar that has fully working convolutional neural networks

A.10 Mitigation of risks

There will probably be some kind of database that has images of children faces, and with the right management of permits that should be fixed. Regarding not having a lot of knowledge in the chosen environment it is assumable since similar tools must have been used during the previous study years. Furthermore, it have been taken into account to put some hours for studying this environment, and if necessary there is an extra margin to learn it. Last, the convolutional newral networks, it is a really famous topic so, some of

the open communities in the Internet should have develop some kind of plugin to work with this, and even if there is not any code for the chosen environment, this other work can be adapted to work in the environment.

A.11 Viability

If as stated with the thesis director, 5 hours a day are spent working in the project, that would made 300 hours, which is in the range of time that should be spent in the project. Even if more time is needed, there is an extra time that can be used without exceeding the maximum time to be spent in the project. So, after analysing the risks, the only one that appears are the restrictions of having images of children, but if an appropiate database is obtained, and the tasks are done on time according to what has been stated in this document, the project is viable and result should be more than satisfactory.

A.12 Estimation of time

Name of task	Esstimated hours	Real hours
Software		
Matlab instalation	2h	2h
Neural Networks		
Get bibliography of NN	2h	3h
Document the theory	50h	70h
Meetings	10h	11h
Monitoring and control	3h	4h
Convolutional Neural Networks		
Get bibliography of CNN	3h	4h
Document the theory	50h	60h
Meetings	15h	15h
Monitoring and control	4h	4h
Facial Expression Recognition		
Get bibliography of Expression Recognition	3h	4h
Document the theory	40h	50h
Meetings	2h	2h
Monitoring and control	3h	3h
Database		
Study of the necessary features	3h	4h
Search for DB	3h	4h
Permissions for DB	2h	2h
Connectivity MySQL and Matlab	5h	10h
Check DB and prepare it for the project	15h	20h
Experiments		
Prepare files for WEKA	20h	20h
Do experiments	10h	15h
Analysis	15h	15h
Meetings	20h	20h
Monitoring and control	5h	5h
Appendices		
Planification	10h	15h
Total	295h	362h

Table A.2: Total hours of the project

B. APPENDIX

Matlab Files

B.1 Check Database .m file

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script to see errors in DB
%
% Column used is added in sortable
% Column usedones is added in spreadsheet
% All the elements in spreadsheet are checked to
% see if all the elements are used.
% Both colums are deleted after that
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all;
clear all;
clc;

db_name = 'faces_db';

% Specific data
sornewcolname = {'used'};
```

```
sortablename = 'sortable';

% General data
spnewcolname = {'usedones'};
sptablename = 'spreadsheet';

% Connect to database
conn = database(db_name, 'root', 'root');
if (isempty(conn.Message))
    disp('Database connected');
else
    disp('Cannot connect database');
    disp(conn.Message);
    return; % stop running
end

% Import whole table of info of each image
sql = sprintf('SELECT * FROM `%s`', sortablename);
incurs = exec(conn, sql);
% Import data into workspace from cursor object
incurs = fetch(incurs);
% Get data from database
indata = get(incurs, 'Data');

% Add new column in img detailed info table
sql2 = sprintf('ALTER TABLE `%s` ADD `%s` VARCHAR(10)',
   sortablename, sornewcolname{1,1});
exec(conn, sql2);
fprintf('In table %s: column `%s` ADDED\n', sortablename,
   sornewcolname{1,1});

[sorsizey, sorsizex] = size(indata);

% Import whole table of general data
```

```
51 sql3 = sprintf('SELECT * FROM `%s`', sptablename);
52 incurs2 = exec(conn, sql3);
53 incurs2 = fetch(incurs2);
54 indata2 = get(incurs2, 'Data');
55
56 % Add new column in general info table
57 sql4 = sprintf('ALTER TABLE %s ADD `%s` VARCHAR(10)', sptablename,
       spnewcolname{1,1});
58 exec(conn, 'ALTER TABLE spreadsheet ADD usedones VARCHAR(10)');
59 fprintf('In table %s: column `%s` ADDED\n', sptablename,
      spnewcolname{1,1});
60
61 [spsizey, sizex] = size(indata2);
62
63 % Go through the elements in the general table, check if they are
      in the
64 % specific table. Puts in the column 'Yes' if they are used, 'No
      if not
65 for i=1:spsizey
66     %fprintf('i: %d\n', i);
67     participantid = indata2{i,5};
68     %fprintf('aux: %s\n', participantid);
69
70     sql5 = sprintf('SELECT `Participant` FROM %s WHERE `
      Participant`=''%s''', sortablename, participantid);
71     curres = exec(conn, sql5);
72     curres = fetch(curres);
73     resdata = get(curres, 'Data');
74
75     [ressizey, ressizex] = size(resdata);
76     %fprintf('num elems: %d\n', ressizey);
77     if (ressizey>0)
78         used = 'Yes';
79     else
```

```
80          used = 'NO';
81      end;
82
83      sqlStr6 = sprintf('WHERE `participant`=''%s''', participantid)
        ;
84      update(conn, sortablename, sornewcolname, cellstr('Yes'),
        sqlStr6);
85      sqlStr7 = sprintf('WHERE `participant-ID`=''%s''',
        participantid);
86      update(conn, sptablename, spnewcolname, cellstr(used), sqlStr7
        );
87  end;
88
89  % Recount the elements in general table to check if sizes match
90  sqlStr8 = sprintf('SELECT COUNT(*) FROM `%s` WHERE `%s`=''Yes''',
        sptablename, spnewcolname{1,1});
91  fincurs = exec(conn, sqlStr8);
92  fincurs = fetch(fincurs);
93  result = get(fincurs, 'Data');
94
95  if (result{1,1}==157)
96      fprintf('%s table is OKAY\n', sptablename);
97  else
98      fprintf('ERROR in %s\n', sptablename);
99  end;
100
101 % Recount the elements in specific table to check if sizes match
102 sql9 = sprintf('SELECT COUNT(*) FROM %s WHERE `%s`=''Yes''',
        sortablename, sornewcolname{1,1});
103 fincurs2 = exec(conn, sql9);
104 fincurs2 = fetch(fincurs2);
105 result2 = get(fincurs2, 'Data');
106
107 if (result2{1,1}==1192)
```

```
108     fprintf('%s table is OKAY\n', sortablename);
109 else
110     fprintf('ERROR in %s\n', sortablename);
111 end;
112
113 % delete created columns in both tables
114 sql10 = sprintf('ALTER TABLE %s DROP %s', sortablename,
       sornewcolname{1,1});
115 exec(conn, sql10);
116 fprintf('In table %s: column `%s` REMOVED\n', sortablename,
       sornewcolname{1,1});
117 sql11 = sprintf('ALTER TABLE %s DROP %s', sptablename,
       spnewcolname{1,1});
118 exec(conn, sql11);
119 fprintf('In table %s: column `%s` REMOVED\n',sptablename,
       spnewcolname{1,1});
120
121
122 % close all cursor connections
123 close(incurs);
124 close(incurs2);
125 close(curres);
126
127 close(fincurs);
128 close(fincurs2);
129 fprintf('All cursor connections closed\n');
130
131 % close database connection
132 close(conn);
133 fprintf('Database connection closed\n');
```

B.2 Check All Images .m file

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script to relate folders to identifiers

% Gives each line in specific info table the real
% file name in the sessions folder which contains
% all the image files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all;
clear all;
clc;

db_name = 'faces_db';

% Specific data
sortablename = 'sortable';

% General data
sptablename = 'spreadsheet';

% New column with the name of the files in the folders
colname = {'Real Filename'};

% Path to the sessions folders
address = 'C:\Users\Usuario\Desktop\DB\IMPORTAR_2\sessions';

% Connect to database
conn = database(db_name, 'root', 'root');
if (isempty(conn.Message))
    disp('Database connected');
else
```

```
32      disp('Cannot connect database');
33      disp(conn.Message);
34      return; % stop running
35  end
36
37  sql = sprintf('ALTER TABLE %s ADD `%s` varchar(120)', sortablename
        , colname{1,1});
38  exec(conn, sql);
39  fprintf('In table %s: column `%s` ADDED\n', sortablename, colname
        {1,1});
40
41  [m n] = size(sortablename);
42
43  % Get folder names of directories
44  f = fullfile(address);
45  mydir = dir(f);
46  directoryNames = {mydir([mydir.isdir]).name};
47  directoryNames = directoryNames(~ismember(directoryNames,{'.','..'
        }));
48
49  colnameaux = strcat('`', colname, '`');
50
51  % Go through the folders to get the names of the images inside it
52  for i=1:numel(directoryNames)
53      fprintf('Directory: %s\n', directoryNames{1,i});
54      auxfile = directoryNames{1,i};
55
56      % Get files that end with .jpg
57      foldaddress = strcat(address, '\', auxfile, '\*.jpg');
58      f2 = fullfile(foldaddress);
59      foldir = dir(f2);
60
61      % Get the participant id from the general info table
```

```
62      sql2 = sprintf('SELECT `participant-ID` FROM %s WHERE `session
    -id`=%s', sptablename, auxfile);
63      curs = exec(conn, sql2);
64      curs = fetch(curs);
65      participant = get(curs, 'Data');
66      participant = participant{1,1};
67
68      % Format the participant string to make queries
69      coma = '''';
70      participant = strcat(coma, participant, coma);
71
72       % Go through the images in the current folder
73      for filej = foldir'
74          imname = filej.name;
75          fprintf('File: %s\n', imname);
76          % Get common part of the im name
77          [token, remain] = strtok(imname, '-');
78          remainaux = remain(2:length(remain)-4);
79
80          % Format remainaux(filename) for query
81          coma = '''';
82          filename = strcat(coma, remainaux, coma);
83
84          % Get the IRT code to know which line has to be given the
    file name
85          sql3 = sprintf('SELECT `IRT Code` FROM %s WHERE `
    Participant`=%s AND LOCATE(%s, `Filename`)>0', sortablename,
    participant, filename);
86          code = exec(conn, sql3);
87          code = fetch(code);
88          codedata = get(code, 'Data');
89
90          sql4 = sprintf('WHERE `IRT Code`=%d', codedata{1,1});
91
```

```
92
93          update(conn, sortablename, colnameaux, cellstr(imname),
    sql4);
94          close(code);
95      end;
96  end;
97
98  % Close all data connections
99  close(curs);
100 fprintf('All cursor connections closed\n');
101
102 % Close database connection
103 close(conn);
104 fprintf('Database connection closed\n');
```

B.3 Crop images .m file

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  % Script to crop all the original images, create
3  % the same folder structure and add the new
4  % information to the specific information table
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7  close all;
8  clear all;
9  clc;
10
11 db_name = 'faces_db';
12
13 % Specific data
14 sortablename = 'sortable';
15
```

```
% General data
sptablename = 'spreadsheet';

% Column with the name of the files in the folders
colname = {'Real Filename'};

% New column with the cropped image file names
cropname = {'Cropped Filename'};

% Path to the sessions folders
address = 'C:\Users\Usuario\Desktop\DB\IMPORTAR_2\sessions';
cropaddress = strcat(address, '_cropped');

% Connect to database
conn = database(db_name, 'root', 'root');
if (isempty(conn.Message))
    disp('Database connected');
else
    disp('Cannot connect database');
    disp(conn.Message);
    return; % stop running
end

% Create folder for cropped images
mkdir(cropaddress);

sql = sprintf('SELECT * FROM %s', sortablename);
sorcurs = exec(conn, sql);

% Import data into workspace from cursor object
sorcurs = fetch(sorcurs);

% Get data from database
sordata = get(sorcurs, 'Data');
```

```
50
51  [sorsizey sorsizex] = size(sordata);
52
53  sql2 = sprintf('SELECT * FROM %s', sptablename);
54  spcurs = exec(conn, sql2);
55  spcurs = fetch(spcurs);
56  spdata = get(spcurs, 'Data');
57
58  [spsizey spsizex] = size(spdata);
59
60  sql3 = sprintf('ALTER TABLE %s ADD `%s` VARCHAR(100)',
       sortablename, cropname{1,1});
61  exec(conn, sql3);
62
63  coma = '''';
64
65  auxcropname = strcat('`', cropname, '`');
66
67  %go throught the images to create the crop images
68  for i=1:spsizey
69      fprintf('i: %d\n', i);
70      foldername = spdata{i,1};
71      participant = spdata{i,5};
72      participantfsql = strcat(coma, participant, coma);
73
74      % Get the full name of the original file
75      sql3 = sprintf('SELECT `Real Filename` FROM %s WHERE `
       Participant`=%s', sortablename, participantfsql);
76      filescurs = exec(conn, sql3);
77      filescurs = fetch(filescurs);
78      filesdata = get(filescurs, 'Data');
79
80      % Create folders for the images following original folder
       structure
```

```
81      imcropfoldaddress = strcat(cropaddress, '\', int2str(
        foldername));
82      mkdir(imcropfoldaddress);
83
84      sizefilesdata = numel(filesdata);
85
86      % For each image in the folder i
87      for j=1:sizefilesdata
88          fprintf('j: %d\n', j);
89          imaddress = strcat(address, '\', int2str(foldername), '\',
        filesdata{j,1});
90          im = imread(imaddress);
91
92          % Convert to single image
93          im_ = im2single(im);
94          fprintf('  Single image: %s\n', filesdata{j,1});
95
96          % Detect face and crop image
97          faceDetect = vision.CascadeObjectDetector();
98          bbox = step(faceDetect, im_);
99
100         % If the image is too big, bbx is empty, so reduce the
        size of the
101         % image
102         while (numel(bbox)<1)
103             im_ = imresize(im_, 0.95);
104             bbox = step(faceDetect, im_);
105         end;
106
107         % Rectify the bboxes that have more than one answer
108         [ysize xsize] = size(bbox);
109         if (ysize>1)
110             bbox = bbox(ysize, :);
111         end;
```

```
112
113         % Crop image of face
114         face = imcrop(im_, bbox);
115
116         auxfaceaddress = strcat(imcropfoldaddress, '\crpd',
    filesdata{j,1});
117         newname = strcat('crpd', filesdata{j,1});
118
119         % Create new jpg image
120         imwrite(face, auxfaceaddress);
121         fprintf('  Image writed: %s\n', newname);
122
123         auxfilesdata = strcat(coma, filesdata{j,1}, coma);
124         sql4 = sprintf('WHERE `Real Filename`=%s', auxfilesdata);
125
126         % Add the new filename to the table
127         update(conn, sortablename, auxcropname, cellstr(newname),
    sql4);
128         fprintf('  After update\n');
129
130         %clear variables
131         clear imaddress im im_ y x z bbox ysize xsize face
    auxfaceaddress newname auxfilesdata sql4;
132     end;
133
134     % Close curs to folder
135     close(filescurs);
136 end;
137
138 % close database connection
139 close(conn);
140 fprintf('Database connection closed\n');
```

B.4 Save features

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script to save all features after applying the
% NN to all the images

% It creates a new table with 2 columns; 1 to
% relate it ith the rest of the tables, and another
% one with the name of the variable containing
% the variable with the features.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all;
clear all;
clc;

% Name of the database
db_name = 'faces_db';

% Specific data table
sortablename = 'sortable';

% General data table
sptablename = 'spreadsheet';

% New table name, contains results
newtable = 'results';

% Column contains name of the results variable
name = 'Result Variable';

% Column name to identify new table in specific table
name2 = 'result-id';
```

```
32
33 % Column of cropped images
34 cropcolum = 'Cropped Filename';
35
36 % Path to the cropped sessions folders
37 address = 'C:\Users\Usuario\Desktop\DB\IMPORTAR_2\sessions_cropped
       ';
38
39 % Connect to database
40 conn = database(db_name, 'root', 'root');
41 if (isempty(conn.Message))
42     disp('Database connected');
43 else
44     disp('Cannot connect database');
45     disp(conn.Message);
46     return; % stop running
47 end
48
49 % Handle the identifier and create table
50 % Create address of the variables column in new table
51 % Add identifier of the new table in the specific table
52 sql = sprintf('CREATE TABLE %s (`%s` VARCHAR(50) NULL, `%s`
       VARCHAR(100) NULL)', newtable, name2, name);
53 exec(conn, sql);
54
55 % Get data of images to loop over them
56 sql2 = sprintf('SELECT `session-id`,`participant-ID` FROM %s',
       sptablename);
57 curs = exec(conn, sql2);
58 curs = fetch(curs);
59 sprdata = get(curs, 'Data');
60
61 [y x] = size(sprdata);
62
```

```
63 % setup MatConvNet
64 run matlab/vl_setupnn
65
66 % load the pre-trained CNN
67 net = load('vgg-face.mat') ;
68
69 % Loop over the images
70 for i=1:y
71     fprintf('i: %d\n', i);
72     sql3 = sprintf('SELECT `%s`,`Filename` FROM %s WHERE `
   Participant`=%s', cropcolum, sortablename, strcat('''', sprdata
   {i,2}, ''''));
73     curs2 = exec(conn, sql3);
74     curs2 = fetch(curs2);
75     sordata = get(curs2, 'Data');
76
77     [y2 x2] = size(sordata);
78     foladdress = strcat(address, '\', int2str(sprdata{i,1}));
79
80     for j=1:y2
81         fprintf('j: %d\n', j);
82         imaddress = strcat(foladdress,  '\', sordata{j,1});
83
84         im = imread(imaddress);
85         % note: 255 range
86         im_ = single(im) ;
87         % resize to 224x224 (expected input size in the network)
88         im_ = imresize(im_, net.meta.normalization.imageSize(1:2))
    ;
89         % normalize
90         im_ = bsxfun(@minus,im_,net.meta.normalization.
   averageImage) ;
91         % apply net
92         res = vl_simplenn(net, im_) ;
```

```
93          % extract 4K features
94          feat4K = reshape( res(end-2).x, 4096, 1);
95          % apply L2 norm to features
96          features = feat4K / norm( feat4K, 2 );
97          % Convert into column
98          trfeature = features';
99
100         % Save the feature column in the table and in the cropped
    image forlder
101         varname = strcat(sordata{j,2}, '_var.mat');
102         fprintf('Varname %s', varname);
103         varaddress = strcat(foladdress,  '\', varname);
104         save(varaddress, 'trfeature');
105
106         % Insert the identifier in res table
107         fastinsert(conn, newtable, cellstr(strcat('`', name2, '`')
    ), cellstr(sordata{j,2}));
108
109         % Insert variable name in res table
110         sql4 = sprintf('WHERE `%s`=%s', name2, strcat('''',
    sordata{j,2}, ''''));
111         update(conn, newtable, cellstr(strcat('`', name, '`')),
    cellstr(varname), sql4);
112     end;
113
114     % Close pointer
115     close(curs2);
116 end;
117
118 % Close pointer
119 close(curs);
120 fprintf('All cursor connections closed\n');
121
122 % Close database connection
```

```
123 close(conn);
124 fprintf('Database connection closed\n');
```

Bibliography

[Ambady and Rosenthal, 2005] Ambady, N. and Rosenthal, R. (2005). Thin slices of expressive behavior as predictors of interpersonal consequences: A meta-analysis. *Psychological Bulletin*, pages 256–274.

[apiexamples.com, 2015] apiexamples.com (2015). Tanh function.

[Bartlett et al., 2003] Bartlett, M., Littlewort, G., Braathen, B., Sejnowski, T., and Movellan, J. (2003). A prototype for automatic recognition of spontaneous facial actions. *Advances in Neural Information Processing Systems*, pages 1271–1278.

[Bartlett et al., 2005] Bartlett, M., Littlewort, G., Frank, M., Lainscsek, C., Fasel, I., and Movellan, J. (2005). Recognizing facial expression: machine learning and application to spontaneous behavior. *Proc. IEEE Int Conf. Computer Vision and Pattern Recognition*, pages 568–573.

[Bartlett et al., 1996] Bartlett, M., Viola, P., Sejnowski, T., Golomb, B., Larsen, J., Harger, J., and Ekman, P. (1996). Classfying facial actions. *Advances in Neural Information Processing Systems 8*, pages 823–829.

[Buranajun et al., 2007] Buranajun, P., Sasananan, M., and S., S. (2007). Prediction of product design and development success using artificial neural network.

[Cohn et al., 2004a] Cohn, J., Reed, L., Ambadar, Z., Xiao, J., and Moriyama, T. (2004a). Automatic analysis and recognition of brow actions in spontaneous facial behavior. *Proc. IEEE Int Conf. Systems, Man and Cybernetics*, pages 610–616.

[Cohn et al., 2004b] Cohn, J., Reed, L., Ambadar, Z., Xiao, J., and Moriyama, T. (2004b). Automatic analysis and recognition of brow actions in spontaneous facial behavior. *Proc. IEEE Int Conf. Systems, Man and Cybernetics*, pages 610–616.

[Demuth and Beale, 1993] Demuth, H. and Beale, M. (1993). Neural network toolbox for use with matlab.

[Donato et al., 1997] Donato, G., Bartlett, M., Hager, J., Ekman, P., and Sejnowski, T. (1997). Classifying facial actions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 974–089.

[Donato et al., 1999] Donato, G., Bartlett, M., Hager, J., Ekman, P., and Sejnowski, T. (1999). Classifying facial actions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 974–989.

[Ekman, 1992] Ekman, P. (1992). Facial expressions of emotion: New findings, new questions. *Psychological science*, pages 34–38.

[Ekman et al., 2002] Ekman, P., Friesen, W., and Hager, J. (2002). Facial action coding system, a human face.

[Friesen and Ekman, 1976] Friesen, W. V. and Ekman, P. (1976). Pictures of facial affect. *Consulting psychologists press*.

[Gokturk et al., 2002] Gokturk, S., Bouguet, J., Tomasi, C., and Girod, B. (2002). Model-based face tracking for view independent facial expression recognition. *Proc. IEEE Int Conf. Face and Gesture Recognition*, pages 272–278.

[Gross et al., 2006] Gross, R., Matthews, I., and Baker, S. (2006). Active appearance models with occlusion. *J. Image and Vision Computing*, pages 593–604.

[Huang et al., 2015] Huang, G., Huang, G., Song, S., and You, K. (2015). Trends in extreme learning machines: A review. *Nural Networks*, 61:32–48.

[Kanade et al., 2000] Kanade, T., Cohn, J., and Tian, Y. (2000). Comprehensive database for facial expression analysis. *Proc. IEEE Int Conf. Face and Gesture Recognition*, pages 46–53.

[LeCun and Yann, 1998] LeCun and Yann (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEE 86.11*, pages 2278–2324.

[Lien et al., 1998] Lien, J., Kanade, T., Cohn, J., and Li, C. (1998). Subtly different facial expression recognition and expression intensity estimation. *Proc. IEEE Int Conf. Computer Vision and Pattern Recognition*, pages 853–859.

[LoBue, 2014] LoBue, V. (2014). The child affective facial expression (cafe) set.

[Lobue and Thrasher, 2015] Lobue, V. and Thrasher, C. (2015). The child affective facial expression (cafe) set: Validity and reliability from untrained adults. *Frontiers in psychology*, 5:1532.

[Lyons et al., 1999] Lyons, M., Budynek, J., and Akamatsu, S. (1999). Automatic classification of single facial images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 1357–1362.

[Moghaddam and Pentland, 1997] Moghaddam, B. and Pentland, A. (1997). Probabilistic visual learning for object recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 696–710.

[Pantic and Barlett, 2007] Pantic, M. and Barlett, M. (2007). Machine analysis of facial expressions. *I-Tech Education and Publishing*, pages 377–416.

[Pantic and Patras, 2005] Pantic, M. and Patras, I. (2005). Detecting facial actions and their temporal segments in nearly frontal-view face image sequences. *Proc. IEEE Int Conf. on Systems, Man and Cybernetics*, pages 3358–3363.

[Pantic et al., 1998] Pantic, M., Rothkrantz, L., and Koppelaar, H. (1998). Automation of non-verbal communication of facial expressions. *Proc. Conf. Euromedia*, pages 86–93.

[Pantic et al., 2005] Pantic, M., Sebe, N., Cohn, J., and Huang, T. (2005). Affective multimodal human-computer interaction. *Proc. ACM Int Conf. on Multimedia*, pages 669–676.

[Parkhi et al., 2015] Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015). Deep face recognition. *British Machine Vision Conference*.

[Picard, 1997] Picard, R. (1997). Affective computing. *MIT Press*.

[Raj, 2016] Raj, B. (2016). Convolution example.

[Rowley et al., 1998] Rowley, H., Baluja, S., and Kanade, T. (1998). Neural network-based face detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 23–38.

[Schneiderman and Kanade, 2000] Schneiderman, H. and Kanade, T. (2000). A statistical model for 3d object detection applied to faces and cars. *Proc. Conf. Computer Vision and Pattern Recognition*, pages 746–751.

[Sun and Jin, 2016] Sun, W. and Jin, Z. (2016). *Advances in Face Image Analysis: Theory and Applications*. Bentham Books.

[Theodoridis and Koutroumbas, 2008] Theodoridis, S. and Koutroumbas, K. (2008). *Pattern Recognition*. Academic Press.

[Turing Finance, 2014] Turing Finance (2014). 10 misconceptions about neural networks.

[Valstar and Pantic, 2006] Valstar, M. and Pantic, M. (2006). Fully automatic facial action unit detection and temporal analysis. *Proc. IEEE Int Conf. Computer Vision and Pattern Recognition*, page 149.

[Valstar et al., 2006] Valstar, M., Pantic, M., Ambadar, Z., and Cohn, J. (2006). Spontaneous vs. posed facial behavior: Automatic analysis of brow actions. *Proc. ACM Int Conf. Multimodal Interfaces*.

[Vedaldi, 2015] Vedaldi, A. (2015).

[Vedaldi and Lenc, 2015] Vedaldi, A. and Lenc, K. (2015). Matconvnet – convolutional neural networks for matlab. In *Proceedings of the ACM Int. Conf. on Multimedia*.

[Vukadinovic and Pantic, 2005] Vukadinovic, D. and Pantic, M. (2005). Fully automatic facial feature point detection using gabor feature based boosted classifiers. *Proc. IEEE Int Conf. Systems, Man and Cybernetics*, pages 1692–1698.

[Wikimedia Commons, 2008] Wikimedia Commons (2008). The logistic sigmoid function.

[Wikimedia Commons, 2015] Wikimedia Commons (2015). Max pooling.

[Yu et al., 2015] Yu, W.and Zhuang, F., He, Q., and Shi, Z. (2015). Learning deep representations via extreme learning machines. *Neurocomputing*, 149:308–315.

[Zeiler and R., 2011] Zeiler, M. D. and R., F. (2011). *Visualizign and understanding convolutional networks*.

[Zhous et al.,] Zhous, S., Chen, Q., and Wang, X. Active deep learning method for semi-supervised sentiment classification. *Neurocomputing*, 120:536–546.