BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION COMPUTER SCIENCE

DIPLOMA THESIS

Multi-feature Facial Emotion Recognition: combining Convolutional Neural Networks with Optical Flow based Action Unit detection

Supervisor
Lecturer PhD. Borza Diana-Laura

Author
Petcu Carina

2022

UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ

LUCRARE DE LICENŢĂ

Recunoașterea Emoțiilor Faciale prin combinarea Rețelelor Neuronale Convoluționale și Analiza Fluxului Optic pentru detectarea Unităților de Acțiune

Conducător științific
Lector dr. Borza Diana-Laura

Absolvent
Petcu Carina

2022

ABSTRACT

In the recent years, the problem regarding the mental health has really been put into light. The current situation in regards to the ongoing pandemic has lead to an increase in the number of people needing therapy. During a therapy session, the psychologist cannot focus only on the facial emotions that the patient is presenting, which can sometimes be crucial in determining the true feelings of a person. The current paper aims to help the therapists in spotting and deciphering the facial expressions that a patient is showing.

The Facial Emotion Recognition area has been progressing in the recent years, various methods of detection being implemented. This work proposes a multi-feature recognition method, by using Convolutional Neural Networks and the Action Units defined by Paul Ekman.

The CNN based method determines the emotions in a static way, by considering only a certain frame from a video for recognizing the emotions. It makes use of pre-trained models like EfficientNetB0, MobileNetV2 and VGGFace, by Fine-Tuning these architectures to maximize the accuracy of the model.

The second method focuses on dynamic facial features. This means that the detection is done by determining the emotion from how the facial movements changed between two consecutive frames. In order for the recognition to be done, Paul Ekman's Action Units are used, as well as the Optical Flow and Landmark Points.

To determine the efficiency of the algorithms, both time and accuracy wise, multiple experiments were performed. Initially, several CNN models are determined, out of which an EfficientNetB0 based one is chosen, having an accuracy of 82% after being fine-tuned.

After the use CNN model is decided, a comparison between the two developed methods is done, considering videos from DISFA dataset. These tests outlined how the two methods can differ in recognizing emotions, considering the way of processing the input data. Time-wise, they performed similarly, with a rate of around 4.5 frames / second. Computationally-wise, some similarities and differences are observed, like the tendency of both methods to determine a specific emotion (angry for CNN based and neutral for Conventional), as well as observing that both the methods determine the disgust and happiness state in a similar way.

These two algorithms have been integrated in a web application to make the usage easier for the therapists. Besides being able to check statistics for a given video emotion-wise, the users can also use the application as a way of centralizing all their patients data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Motivation

Paul Ekman mentions that the emotions change how we see the world and how we interpret the actions of others[12]. Facial expressions are one of the most used ways of revealing internal thoughts and emotions, by non-verbal means. Understanding this unspoken way of expressing feelings could be of help to therapists, teachers, and even ordinary people during their day to day duties.

In 1967, doctor Paul Ekman worked with clinical cases in which patients lied about their emotional state. He studied patients who claimed they were not depressed and later committed suicide. Upon examining films of the patients in slow motion, the micro facial expressions were spotted, which revealed strong negative feelings the patients were trying to hide. Therefore, six expressions were defined by Ekman: anger, disgust, fear, happiness, sadness and surprise, where the neutral state could also be added[13]. They are the basic emotional expressions, universal among human beings.

The facial emotion recognition (FER) topic has been in development for years. In order for the information supplied by the facial expression to be analyzed, the Facial Action Coding System (FACS) model[14] and categorical model are used widely. FACS is the most objective and comprehensive coding system in behavioral scenes. The facial expressions are divided in regards to 46 component movements by a human decoder, divisions which correspond to the individual facial muscles.

The World Health Organization mentioned in March 2022 that the COVID-19 pandemic triggered an 25% increase in the prevalence of anxiety and depression worldwide, which lead also to an increase in the number of people going to a therapist. During a session the therapist cannot only concentrate on the emotions that a person is showing, which are very important considering Paul Ekman's research. Therefore, having an application that could provide the psychologist an accurate result of all the emotions shown during a session could lead to a more accurate analysis of how a patient is feeling.

1.2 Objectives

The aim of this paper is to provide a solution to the psychologists in analyzing the emotions shown by a patient during a therapy session, in an automated way. But the solution is not limiting only to the statistical part, it is also integrated in a web application to make its usage easier.

The most important objective of this paper is to create a reliable way of detecting the facial emotions that a person is showing. The reliability can be measured in regards to the accuracy of the proposed solution in detecting emotional states. Moreover, the computational speed should be fast, as therapy sessions are usually one hour long. The time taken for processing a video of this length should be optimized such that the therapists should not wait long for the statistics.

Considering all the previous work Paul Ekman has done in regards to this matter, another objective is to create a solution based on his work on Facial Action Units and how they can be interpreted to result the emotion on a person's face. But as the Computer Vision area of interest has been developing recently, another objective is to have another solution of the proposed matter, but using Convolutional Neural Networks to detect the emotions. Furthermore, these two methods could be examined and compared, as a way to determine which one is the most efficient both computationally and time wise.

The current paper aims to create an accessible, easy to use and intuitive web application for the therapists. This application could be used as a way of storing all the patients data in a single place, as well as getting the before mentioned statistics.

1.3 Original contributions

The Facial Emotion Recognition area has been in development for several years. Considering the objectives that are set in the previous section, the current paper brings into light some original contributions in regards to this matter.

First of all, it presents a multi-feature Facial Emotion Recognition algorithm, which can be divided into two methods, of different types. One focuses on detecting the emotions from frames, while the other one is a dynamic method, considering two consecutive frames. The first way of detecting is CNN based, using Transfer Learning and Fine Tuning during the training of the model. The second method is based on the work done by Paul Ekman in regards to Action Units, the detection being done by using Optical Flow and Landmark Points.

Another original contribution is the comparison between the two mentioned methods, from a time complexity point of view, as well as the detected emotions both from a single frame and from consecutive ones.

Lastly, another original contribution is the development of an web application, allowing the users (therapists) to upload videos of patients from sessions and to get statistics of the emotions shown, after manually setting some parameters like the preferred detection method or the number of frames that should be taken into account.

1.4 Thesis Structure

The current paper is divided into six chapters, including the current one, with every chapter tackling a different matter into solving the before mentioned objectives.

Chapter 2 presents an overview of all the development that has been done until the present in regards to Facial Emotion Recognition. It begins with a brief overview of the history of emotions. It continues with the history of the automated way of recognizing emotions, taking into account both Conventional methods, as well as Deep-Learning procedures.

Chapter 3 sets the theoretical foundations needed for the proposed solution. Two sections are taken into account, the Convolutional Neural Networks and the two methods of determining the Optical Flow.

Chapter 4 details the proposed solution for the already set objectives. It starts with an overview of the solution. The Facial Emotion Recognition solutions are being detailed after, considering all the modules that compute the statistics. Finally, the application as a whole is presented, considering all the required diagrams and how the elements of the whole system interact with each other.

Chapter 5 presents all the experiments that have been done considering the proposed solution, considering also the implementation. Two types of tests are done, one of them being those considering the model training and getting the best accuracy for the CNN based solution. The second set of tests is done to compare the two proposed methods and to see how each of them behaves.

Chapter 6, which is also the Conclusions, presents an overview of the proposed solutions and their results in comparison to the objective that are set in this Chapter. Moreover, future developments to both the detectors and the web application are mentioned.

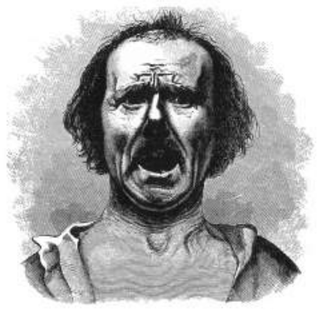
Chapter 2

State of the art

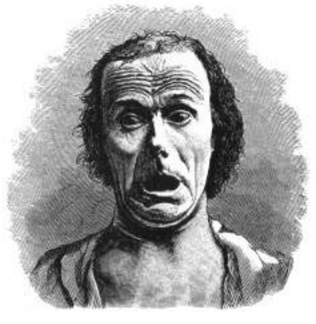
This section will address the work that has already been done in regards to Facial Emotion Recognition. But before jumping to the automated part of this area of interest, some details in regards to the history of human emotions could be mentioned.

2.1 History of emotions

Charles Darwin presented the basic model of emotions in the book published in 1872, *The Expression of Emotions in Man and Animals*[15]. He affirmed that a sentiment involves many systems, such as facial expression, behavioural response and physical responses. Moreover, Darwin was the first one to divide the feelings into six emotional states: happiness, sadness, fear, anger, surprise and disgust, as well as defining the variations in the before mentioned emotions. In figure 2.1, two variations of fear are visually presented.



Horror and Agony[15]



Terror[15]

Figure 2.1: Variations of fear

2.2 History of FER

Using Darwin's work, Paul Ekman analyzed the emotional expressions of one being and created the Facial Action Coding System (FACS) model[13]. What Ekman did was to divide the facial musculature movement based on 46 component movements called Action Units(AU) and on their intensity. Using these AUs in different combinations, all the emotions that one person is capable of feeling are coded, considering only the facial expressions, without the tone of the voice, the body language or other human actions that might influence. For example, happiness is described by combining the AU6, which is the cheek raiser, and AU12, which is the lip corner puller, as it is ilustrated in figure 2.2[1]. Paul Ekman and Wallace V. Friesen tackled the way of coding the emotions from the Facial AUs, by getting the result from combining different behaviours and how to interpret them, resulting in the Facial Action Coding System Affect Interpretation Dictionary (FACSAID) [17].

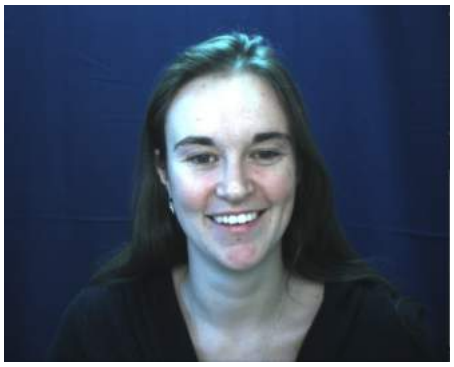


Figure 2.2: Happiness

Considering the substantial interest the humans have shown in regards to studying what other people are expressing by facial means, psychology was not the only domain in which this subject was developed. With the evolution of computer science, the recognition of sentiments has started being automated. Nowadays, there are several types in which the Facial Emotion Recognition can be done.

Firstly, the methods could be divided into applications that are using Artificial Neural Networks or not. Therefore, in the beginning, a classical way was being used. This means that an input image would be received by the application, the face would be extracted, several landmarks would be detected (such as the jaw line, the eyebrows, the nose, the eyes and the lips). Using those landmarks, the Action Units would be detected and the emotion recognized. With the evolution of Convolutional Neural Networks, the ways of extracting the sentiments have changed and lots of research has been done towards this new direction. Nowadays, an input image is also being utilized, but using convolutions various areas of interest from the starting

[1]The image is a frame from one of the videos in the DISFA dataset [16, 11]

frame are being gathered such that the detection is more accurate. This comes with a downside, as it requires more computational power in exchange of the detection being faster.

Secondly, the implemented algorithms for facial emotion recognition could be categorised by the type of the input. In the beginning of automating this process, all the work was done by extracting all the input from images, or frames of videos. Nowadays, with the advancements in Computer Vision, the recognition is done from videos as a whole, in a more dynamic way.

2.3 Conventional FER Procedures

As it was mentioned before, the development of Facial Emotion Recognition algorithms started conventionally, various ways being studied. All of them had in common the detection of face region and for the important landmarks of the face, all the results of the observations encapsulating the recognition of the emotion.

The first papers addressing this matter were mainly focusing on images of front faces. The article published by [18] also tackles the idea of recognizing the facial expressions from static face images, but this time the profile of the face is also taken into consideration. Being one of first scientific papers regarding this matter, it explains the way the Action Unit Recognition is developed. Whereas the dynamism of a facial expression is not denied in this paper, the static images also play a role for obtaining configurational information about them. The face detection is followed by a facial feature one and facial action detection. Overall the result of the experiment led to an average recognition rate between 86.3% and 88%, the authors also outlining the areas of improvement of it.

Ghimire and Lee [19] propose a solution to this problem which involves the usage of feature vectors resulted from individual landmarks. Even though facial image sequences are being used, the algorithm is taking into consideration the way the facial landmarks are evolving over time in different consecutive video frames, by considering the position and the angle of them. After the feature vectors and the pairs of landmarks are extracted, they are normalized considering the first frame of the sequence. Finally, either the multi-class AdaBoost with dynamic time warping or the support vector machine are being used in order to classify the expressions of the detected faces. This algorithm was tested using the CK+ dataset [20] and it reached an accuracy of 95.17% by using multi-class AdaBoost and 97.35% for support vector machines.

The articles mentioned before are only a small part of the way this issue was implemented over time using conventional methods. Every other paper brings something innovative, whether it is in regards of the visual features or of the decision methods.

As visual features, besides the ones already described, worth mentioning is the report written by Gunawan et al. [21], which innovated the recognition of facial emotions by using infrared images, because the usual images (visible light images) are influenced by the lighting. Moreover, Szwoch and Pieniazek [22] did not even use a camera to identify the emotions based on facial expressions. Instead, they made use only of the Microsoft Kinect sensor.

As decision methods, most of the developed algorithm are using the Support Vector Machines (SVMs) - supervised learning models that help in analyzing the data for classification and regression analysis. Besides Ghimire and Lee [19] which is making use also of multi-class AdaBoost, Happy et al. [23] utilise the Principal Component Analysis (PCA). The latest paper utilises a local binary pattern histogram from a global face region instead of the feature vectors, classifying them with PCA.

Finally, the algorithms mentioned before played a big role in the evolution of the automated Facial Emotion Recognition. In comparison with the deep-learning ways, they require less computation power. This is one of the reasons that these type of methods are still being researched. Moreover, they also provide high accuracy. But one cannot simple deny the effectiveness of the deep-learning methods that will be detailed in the next section.

2.4 Deep-Learning FER Procedures

With the evolution of Computer Vision, more ways of detecting the emotions of individuals based on their facial expressions started being developed. Mainly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) started being used. What made the researchers switch from the already established methods to the before mentioned ones was the fact that the images would be the only way of learning, without any physics based computations that have been described in the previous section. Moreover, the CNNs would be used for various works like face recognition, feature extraction, classification of the features and, lastly, emotion recognition. More about the way the CNNs work and how the Computer Vision tasks are developed will be detailed in Chapter 3.

The development of Deep-Learning based FER algorithms also was done progressively. In one of the first papers that address this matter, Jung et al. [24] mention the effort that is done by trying to implement the needed features for recognizing one's emotions by their facial expressions. Therefore, they are proposing a new architecture, which is based on CNNs. The new design is split into two parts, one deep neural network that is extracting the temporal features from image sequences, the other one dealing with the extraction of temporal geometry features, merging the two outcomes by using a joint fine-tuning. By the new integration method, the accuracy of the model exceeded the state of the art at that moment.

Hasani and Mahoor [25] raise the problem of tackling FER when recognizing expressions in videos. The idea behind this work is that every facial expression has a dynamic pattern composed of three phases: set, peak and offset. The proposed solution is 3D Inception-ResNet (3DIR) network, which is an extension to the 2D Inception-ResNet module. 3DIR extracts the spatial relations between frames in a sequence. The relations are taken into consideration by a Long Short-Term Memory (LSTM) [26]. It uses the gathered information to classify the sequences. An addition is the incorporation of facial landmarks, which helps in differentiating facial components and other parts of the face. Moreover, various cross database experiments were inducted, meaning that for every database, the current one is used for testing and the other ones are used for training the model.

The introduction of LSTM played a crucial role in the evolution of FER algorithms. Long Short-Term Memory Ko [27] outlines some advantages that the RNN model has, in comparison to the already existent one. Firstly, he mentions the straightforwardness of LSTM in regards to fine-tuning end-to-end when it is integrated with CNN models. Moreover, the flexibility of the input is also an advantage for this new type of models. Therefore, several LSTM-CNN models have started being developed, including the previously mentioned one. More details about the way LSTM is implemented and its usage are provided in Chapter 3.

Jain et al. [28] submitted a new FER method, which tackles the issue with the brightness and the sudden changes in illumination. The multi-angle optimal pattern-based deep learning (MAOP-DL) is used also to find the proper arrangement of the input by using multi-angle based optimal configurations. This solution has several steps, including the subtraction of the background and isolating the foreground from images, then the texture patterns and the relevant key features of the facial points are extracted, ending with the prediction of the LSTM-CNN model.

The paper published by Akhand et al. [29] is one of the most recent ones. The authors determined that using a shallow CNN in detecting the facial expressions can lead to bad results. Therefore, they propose a solution that uses very Deep CNN modeling through Transfer Learning. Moreover, a new pipeline strategy is proposed, where the fine-tuning of each pre-trained DCNN block follows the training of the dense layer(s).

Another novel method is the one developed by Hossain et al. [30], which proposed a FER system that applies to many areas of interest like healthcare, emotions, social. It follows a basic pattern for this issue, predicting landmark points, analyzing the textures by a CNN and classifying the data. Utilising several CNN models, the results are fused such that the results are improved.

Chapter 3

Theoretical foundations

This section will present in a detailed way the theoretical foundations that were used for developing the application, starting from Convolutional Neural Networks to elements that are used for the conventional detection of emotions, such as Optical Flow.

3.1 Convolutional Neural Networks

Convolutional Neural Networks are, as the name is suggesting, Neural Networks that use convolutions. But before diving deep into this concept, the idea of Neural Networks is presented.

The architecture of ordinary Neural Networks has been heavily influenced by the biological neural systems. The analogy comes from the fact that both neurons, whether the biological or the automated one, have a similar flow in regards to how the information is being managed in order to learn new concepts.

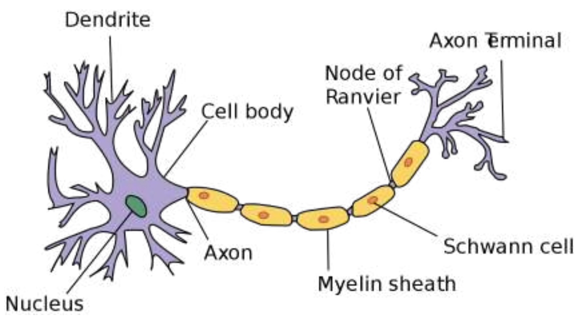


Figure 3.1: Biological Neuron

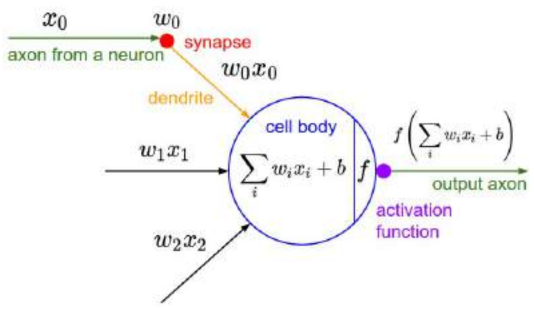


Figure 3.2: Automated Neuron[1]

Figures 3.1 and 3.2 visually represent the before mentioned idea. The usual flow is as follows: dendrites are sending input signals to the neuron, which are carried toward the cell body, reaching the nucleus. The information is carried away passing through the axon, which branches in several terminals. Those are connected to dendrites of other neurons, the information being sent forward. In a similar way, the information (in this case x_0) is the one received from another neuron. A synapse creates the connection between the previous and the current neurons, the information being sent to the current one by the dendrite (w_0 is the synaptic strength, w_0x_0 being the full information). The synaptic strengths, or the weights, are the learnable part of the neuron and control how much influence they have. The automated nucleus sums up all the information received from the automated dendrites ($\sum_i w_ix_i + b$) and using **an activation function f**, the information is sent to the next neuron.

An Artificial Neural Network is compounded of collections of neurons that are connected in an acyclic graph. One thing about this graph is that it does not have cycles, as it would create infinite loops. Figure 3.3 presents an Artifical Neural Network with 2 hidden layers, where all the neurons are interconnected.

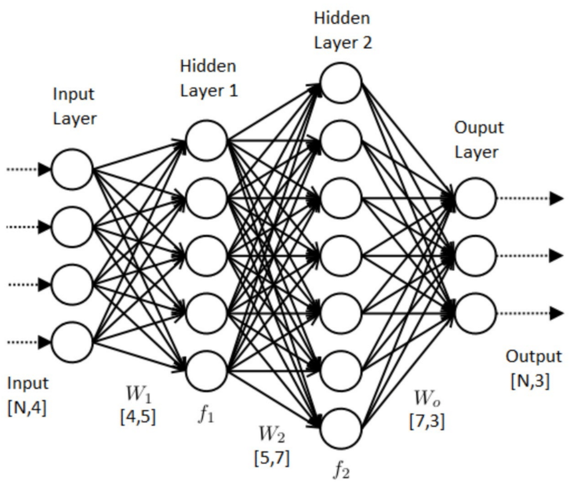


Figure 3.3: Artificial Neural Network

Convolutional Neural Networks are very similar to Artificial Neural Networks, taking into account that both of these two are made up of inter-connected neurons with learnable weights (w_0 from figure 3.2) and biases (b from figure 3.2), each of those receiving an input and sending an output after some computations. One of the big differences is the fact that the Convolutional Neural Networks have as general input several photos, while the output represents the class scores.

It is worth mentioning the fact that ordinary Neural Networks do not scale well in the case of full images. For example, to train a Neural Network using CIFAR-10 [31], a collection of 60000 images with the size 32x32x3 divided equally into 10 classes (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks), using a hidden fully-connected neuron would lead to 32x32x3 = 3072 weights. This is feasible, but the usual image size that Convolutional Neural Networks are using nowadays is around 200x200x3, which means 120000 weights for only one neuron.

Therefore, regular Neural Networks would need very high computational power to be trained.

Convolutional Neural Networks are using the dimensions of an input image such that the computations are done in a more efficient way. This means that instead of having neurons in an acyclic graph, this type of Neural Networks has layers made up of neurons ordered like the input images, in a height x width x depth way, as it presented in figure 3.4.

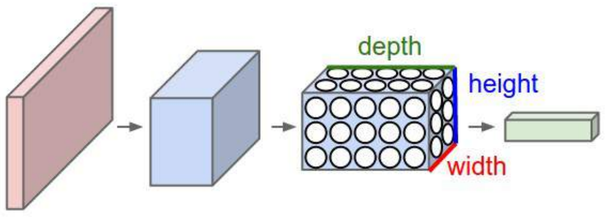


Figure 3.4: Convolutional Neural Network[1]

There are three main types of layers used in building Convolutional Neural Networks: **Convolutional Layer**, **Pooling Layer** and **Fully-Connected Layer**. The API of each layer is straight-forward: from an input 3D volume to an output 3D volume with some differentiable function that may or may not have learnable parameters. These types of layers will be stacked to create a full Convolutional Neural Network architecture.

3.1.1 Convolutional Layer

As the name is suggesting, the Convolutional Layer is the core building block of a Neural Network of this type. It is the one that does most of the computational heavy lifting.

The Convolutional layer's parameters are made of learnable filters. In regards to the dimensions, they are of a smaller size in comparison with the input ones (for example, a typical filter on a first layer might have as dimensions 5x5x3), but they extend to through the full depth of the input volume. But how do these filters work? During the forward pass, every filter is convolved across the width and height of the input volume, dot products between the entries of the filter and the area of the input covered currently by the filter. After all the computations are done, the result will be a 2-dimensional activation map that gives the responses of that filter at every spatial position. The before mentioned computations can be seen in figure 3.5. Moreover, in figure 3.6 it can be seen that each Convolutional Layer is not made up by only one filter, but multiple ones. By stacking the activation maps for each filter, the output volume is resulted.

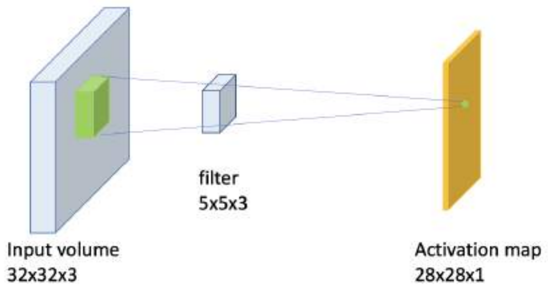


Figure 3.5: Convolutional Layer

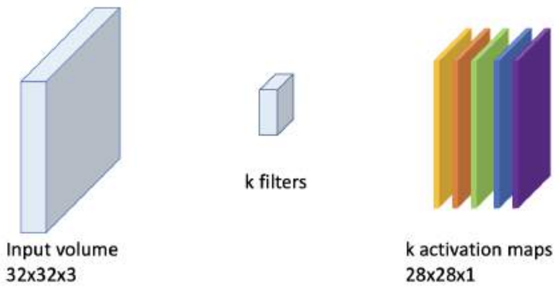


Figure 3.6: Activation Maps

As it was mentioned before, it is impractical to connect the current neurons to all the previous ones. Therefore, by using convolutions, the neurons will be connected only to a local region of the input volume, as it can be seen in figure 3.5, which is called the **receptive field**.

The way each neuron is connected to the previous or the current ones was tackled, but it is also of interest the neurons of the output volume, how they are arranged and what are the dimensions of the new layer. There are three hyperparameters that control the way the output layer is shaped:

- The **depth** of the output volume, which corresponds to the number of filters that are to be applied. Figure 3.6 captures k activation maps from a filter with a given size.

- The **stride** with which the filter is convolved, resulting in smaller output volumes spatially the bigger the stride is. For example, stride 1 (figure 3.7) means that the filters are moved one pixel at a time, while stride 2 means jumping 2 pixels (figure 3.8).

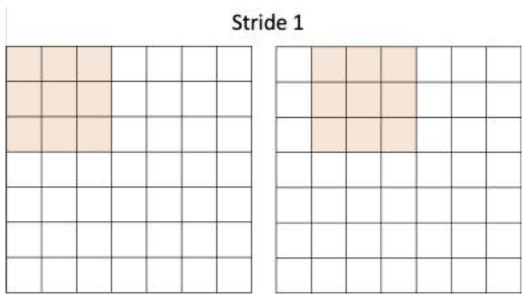


Figure 3.7: Stride 1

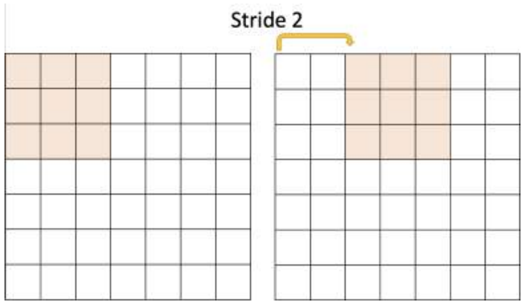


Figure 3.8: Stride 2

- The **padding** is also useful sometimes, depending on the use case, controlling the spatial size of the output volumes. Mostly it is used for preserving the spatial size of the input volume, keeping the height and the width the same.

Taking all the notions presented previously into account, a formula to determine the dimensions of output volume can be computed. Therefore, considering W_0 the width of the input, H_0 the height of the input, D the filter depth (which is the same for the input and for the filter), F the filter size, P the padding and S the stride:

$$W_1 = \frac{W_0 - F + 2 * P}{S} + 1$$
$$H_1 = \frac{H_0 - F + 2 * P}{S} + 1$$

, where W_1 is the width of the output and H_1 is the height of the output.

3.1.2 Pooling Layer

The Pooling Layer is used in order to reduce progressively the spatial size of the layer that is processed. This also results in a reduction of the amount of parameters and computation in the network, controlling also the overfitting. In theory, there are two types of poolings, average and max. In practice, the average pooling is not used as before, since the max one was shown to work better in practice. The Pooling Layer computes max operations on every depth slice of the input and it resizes it spacially, as it is presented in figure 3.9 The most used pooling filters have the size 2x2 with a stride of 2, which discards 75% of the activations. This layer introduces zero learnable parameters and a padding is not usually applied.

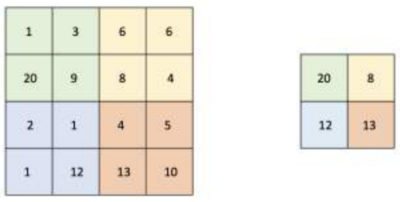


Figure 3.9: Max Pooling Layer

Taking into account the formulas presented at the previous subsection, and the before mentioned notions, the pooling layer produces a volume with the following dimensions:

$$W_1 = \frac{W_0 - F}{S} + 1$$
$$H_1 = \frac{H_0 - F}{S} + 1$$

, where W_1 is the width of the output and H_1 is the height of the output and the depth remains the same.

3.1.3 Fully-connected Layer

This type of layers are the same as the ones mentioned in the beginning, for Neural Networks. All the neurons have full connections to all activations in the previous layer. Moreover, they do not preserve spatial information.

It is worth mentioning the fact that the Fully-Connected Layer can be converted into a Convolutional Layer. For example, in the case that the input volume reaches a smaller size, like 7x7x3, instead of using a Fully-Connected Layer, a Convolutional layer with filter size 7 can be used, having the same result.

3.1.4 Convolutional Neural Networks Architectures

A Convolutional Neural Network is made up of only the three layers defined before, Convolutional Layer, Pooling Layer and Fully-Connected Layer. Usually, the idea of creating an architecture of this kind is to stack a few of the three layers and then to repeat the pattern. In the end, Fully-Connected layers are added, matching with the class scores using the softmax activation function. Therefore, most of the architectures have the following pattern:

$$[[CONV - ReLU] * N - POOL?] * M – [FC - ReLU] * K – FC - softmax$$

The ideas explained before can be seen in figure 3.10. The Convolution - Pooling Layers make up for the Feature Extraction part classifying an input image, while the Fully-Connected layers compose the Classification part of this process.

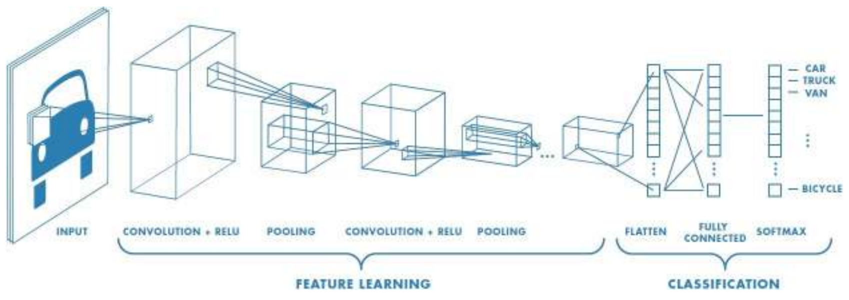


Figure 3.10: CNN Architecture[2]

The most recent Convolutional Neural Networks change this pattern. The trends nowadays are to reduce the filter sizes and to increase the depth of the networks or to avoid using Pooling and Fully-Connected layers, and use only Convolutional layers.

3.2 Optical Flow

Gibson [32] in his work, *The Perception of The Visual World*, changed the way psychology views perception, by asking the question 'How do we see the world as we do?'. This can be considered as the Optical Flow base, being the first to tackle this matter. Moreover, Fleet and Weiss [33] described motion as an intrinsic property of the world and an integral part of the overall visual experience.

Optical Flow is a fundamental concept utilized in most video-processing algorithms, and it is defined as the pattern of apparent motion of objects, surfaces and edges in visual scenes, caused by the relative motion between an observer and a scene. It is a two dimensional vector field, where every vector from the field represents the offset vector, presenting the motion of the points from the first frame to the second one.

When talking about Optical Flow, there are some assumptions to be taken into considerations, like the fact that the neighbouring pixels have similar motion, and the pixel intensities of an object do not change between the first and the second frame. Therefore, taking those suppositions in consideration, the formula to calculate the Optical Flow of a certain point in a frame can be computed. A pixel *I(x, y, t)* is being considered, from the first frame, where *x* and *y* are the coordinates on the two dimensional frame and *t* is the time. This pixel is moving by distance *(*Δ*x,* Δ*y)* in the second frame after Δt time. Therefore, the following changes will result:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

Assuming a small movement, the image constraint at *I(x, y, t)* with Taylor series can lead to, after truncating the higher order terms):

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0$$

After diving by Δt, the equation transforms to:

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0$$

, where V_x and V_y are the *x* and *y* components of the optical flow of the before mentioned *I*. This two unknowns equation cannot be solved by normal means. Every Optical Flow method adds additional conditions in order for the equation to be solvable. Therefore, several methods were developed, out of which two will be discussed in this paper.

3.2.1 Lucas-Kanade method

This method takes into account one of the before-mentioned assumptions, that the neighbouring pixels will have similar motion. Lucas-Kanade method computes the Optical Flow for a sparse feature set. This means that it is computed for certain given points, whether they are corners (mainly detected using the Shi-Tomasi algorithm) or landmark points.

For every point in the sparse set it is taken a 3x3 window around it, meaning that all the 9 points (the initial one included) have the same motion. Therefore, the issue changes to solving a system formed of 9 equations with 2 unknowns. A system like this is over-determined, and applying the least square fit method it is changed to a two equations and two unknowns system.

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \Sigma_i I_x(q_i)^2 & \Sigma_i I_x(q_i) I_y(q_i) \\ \Sigma_i I_y(q_i) I_x(q_i) & \Sigma_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\Sigma_i I_x(q_i) I_t(q_i) \\ -\Sigma_i I_y(q_i) I_t(q_i) \end{bmatrix}$$

, where q_1, q_2, ..., q_n are the points inside the patches, $I_x(q_i)$, $I_y(q_i)$ and $I_t(q_i)$ are the partial derivatives of image *I* with respect to position *x*, *y* with time *t*, evaluated at point q_i.

The only issue right now is the fact that only small motions are taken into account. For larger motions, there is the need for pyramids to be used.

The results of the before mentioned algorithm can be seen in figure 3.11.

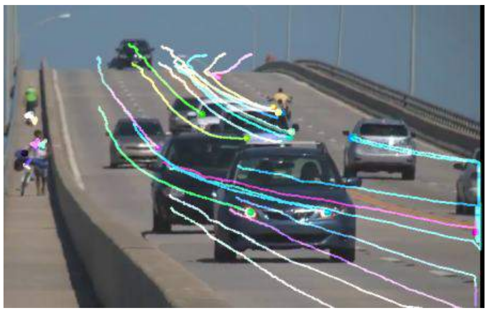


Figure 3.11: Optical Flow using Lucas-Kanade method [3]

3.2.2 Gunnar Farnebäck method

The second Optical Flow method that is tackled in this paper is based on the algorithm developed by Farnebäck [34]. It is an estimation algorithm, based on two frames. For the computation, two steps are required. The first step is to approximate each neighbourhood of both the input frames by quadratic polynomials. The polynomial expansion transform is used such that the algorithm is more systematic. Considering how the polynomial is transforming under translation, a method which estimates the displacements fields from the polynomial expansion coefficients is derived. Some refinements are applied, which leads to the algorithm.

In comparison to the Lucas-Kanade method, this one computes the optical flow for all the points in the frame. The result is a 2-channel array with optical flow vectors *(u, v)*. For a better view of the results, the first element in the vector is used to find the magnitude of the motion, while the second one is used for the direction of the movement. An HSV image is computed using these values, where the Hue is the value of the magnitude, the Saturation is set to maximum by default, for every result and the Value is the direction.

The results of the algorithm can be seen in figures 3.12 and 3.13. The difference between the colours is depending on how the pixels moved and the intensity of the movement.

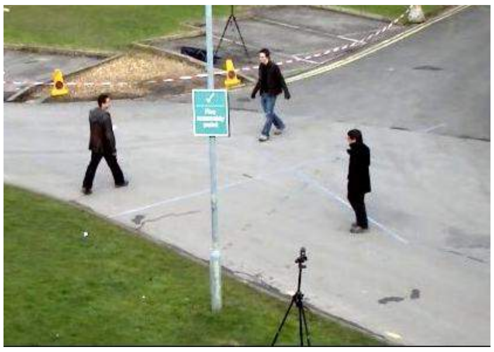


Figure 3.12: Initial frame[3]

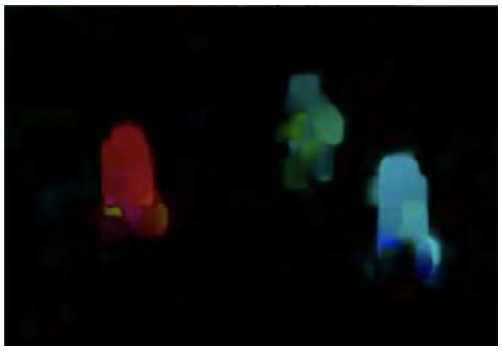


Figure 3.13: Dense Optical Flow[3]

Chapter 4

Proposed solution

This chapter brings into discussion the proposed solution for the discussed matter. It initially tackles an overview of how the application is working, by shallowly presenting the pipeline of the project. Moreover, the methods that handle the Facial Emotion Recognition are described, considering all the modules that compound it. Finally, an in-depth analysis in regards to the application is taken forward, considering the use cases of the proposed solution, the overall architecture and the way the Facial Emotion Recognition algorithms described previously connect with the final web application.

4.1 Solution overview

The current section is mainly focusing on the emotion recognition matter, as it is the starting point of this paper. Figure 4.1 presents the pipeline and the steps needed in order to detect the emotion of each person in a video.

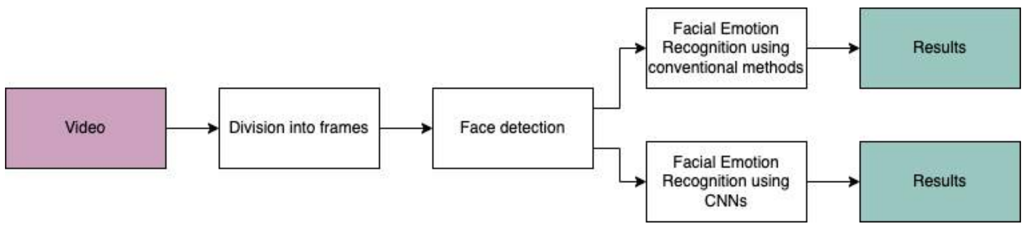

Video
Division into frames
Face detection
Facial Emotion Recognition using conventional methods
Facial Emotion Recognition using CNNs
Results
Results

Figure 4.1: Pipeline

It starts with the input, which is a video of various length. The videos are uploaded from the web application and they need to present people, no matter if they present emotions or not. The second step is to pre-process the current video, by diving it into a specific number of frames. After the video is split, each frame is one again modified, by selecting only the faces from it.

The next step can be divided into two, depending on the chosen method of defining the emotions of the detected faces at the previous step. This paper outlines two ways of feelings recognition, taking into account the previous work that was done previously and presented in Chapter 2. Initially, there is a conventional method that makes use of Facial Landmark Points and Optical Flow, as well as the work done by Paul Ekman in detecting Action Units. Secondly, a method using Convolutional Neural Networks is also proposed. Finally, the results depending on the number of frames are sent back to the user.

The before mentioned ideas are presented in a more detailed way in Section 4.2.

4.2 Facial Emotion Recognition Module

This section takes into discussion the algorithms that represent the center of this current work. All the modules that make part of this are detailed, considering various ways of recognising the facial emotions of a person from videos.

As it was mentioned in Section 4.1, two solutions are proposed, one which is conventional and takes into account the way the facial muscles are moving and one using Convolutional Neural Networks. Even though these two methods vary, the pre-processing of the videos is a common part of the algorithm.

Therefore, the flow follows the one presented in figure 4.1, presenting all the steps needed, starting from the Face Detection part, and outlining in a more insightful way how the Emotion Recognition is implemented.

4.2.1 Face Detection

Following a similar idea to the one for recognizing the facial emotions, several ways of detecting the face were implemented, as a method of finding the one that suited the best this use case.

The Factory Design Pattern, which is a creational design pattern that provides an interface for creating objects in a super-class, but allows sub-classes to alter the type of objects that will be created, is used in order to give the user the chance to choose the preferred method of detecting the faces in a frame. Moreover, it is also used because the execution followed a similar pattern (processing the image, detecting the faces, resizing the faces and returning the results).

As it is observed in Figure 4.2, the base class defines several methods that process the input image, in order to determine the desired result, while the sub-classes that inherit from it override the methods depending on their particularities.

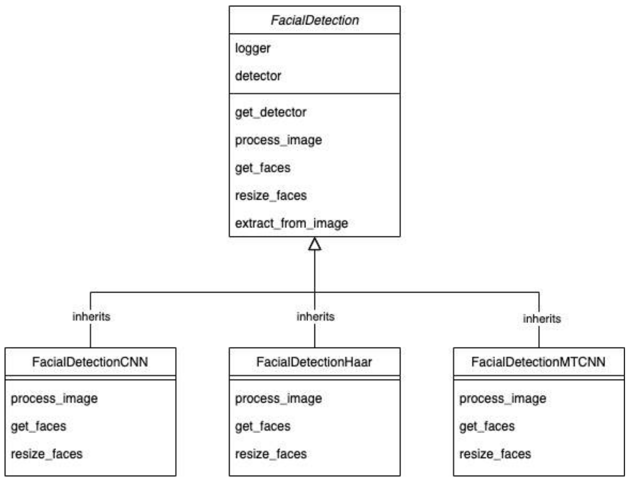


Figure 4.2: Facial Detection Classes

The three implemented facial detectors are using well-known methods, such as **Viola-Jones Face Detection (Haar Cascades)**, **Max-Margin Object Detection** and **Multi-task Cascaded Convolutional Networks**.

Viola and Jones [35] introduced one of the first algorithms used nowadays for visual object detection, **Haar Cascades**. By including in the algorithm three key contributions (”Integral Image”, a learning algorithm based on AdaBoost and combining classifiers in a ”cascade”), they proposed a solution that managed to detect objects at 15 frames per second, which is still one of the fastest algorithms. One downside of using it would be the amount of False Positive objects that are detected.

King [36] brings a contribution to already existing object detection methods, **Max-Margin Object Detection (MMOD)**, which brings an optimization to all the sub-windows already used, also by using Convolutional Neural Networks. Moreover, in order to encourage future research, he made the algorithm open source, on the dlib machine learning toolbox.

Lastly, Zhang et al. [37] tackles the issue of the difficulty in detecting faces in bad illumination conditions or occlusions. Also using Convolutional Neural Networks, a deep cascaded multi-task framework is proposed **(MTCNN)**, having three stages meant to predict the face and landmark location such as eyes, nose and mouth.

All the before mentioned object or face detection algorithms have pros and cons in regards to speed, complexity, as well as accuracy of results. After running several tests, the decision of using the MTCNN detector as the base one is done.

4.2.2 Emotion Detection Components

Using of the methods described in the previous section, all the faces are retrieved. The next step in the flow is to make computations on the faces to determine the emotions that are shown. As it was mentioned, the proposed solution brings forward two methods for detecting the feelings, a conventional one and one idea which is CNN based. Depending on the chosen method, the flow is different and various modules are being used. But before jumping to the detailed part, it is to be noted that both methods detect only a specific range of emotions: **Fear**, **Anger**, **Surprise**, **Disgust**, **Sadness**, **Happiness**, the **Neutral** state also being added. Moreover, the Factory Design Pattern is also used in the development, as it can be seen in figure 4.3.

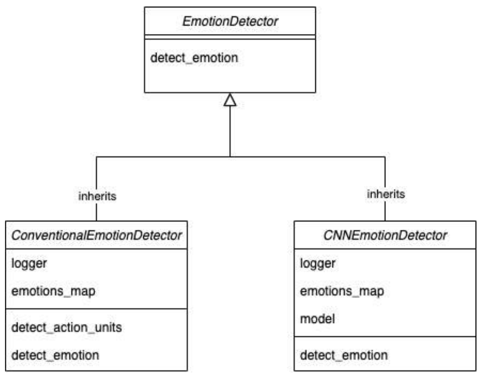


Figure 4.3: Emotion Detector Classes

Therefore, there exists a base interface called **EmotionDetector** that implements only one abstract method, *detect_emotion*. Both the **ConventionalEmotionDetector** and **CNNEmotionDetector** implement this interface, the first one adding one more method, which deals with the recognition of action units. The reason for this implementation, as well as how the two classes work, is explained in the following sections.

Conventional Emotion Detector

The conventional method gathers several notions from various subjects, like psychology, physics, geometry. In figure 4.4, the flow of the method can be seen.

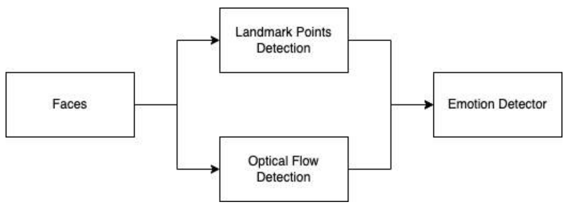


Figure 4.4: Flow of Conventional Emotion Detector

Considering the faces that are extracted during the previous step, the Optical Flow and the Landmark Points are detected. The Optical Flow is marked in comparison with previous frames, while the Landmark Points are found depending on the faces from the current frame. After these elements are computed, they are sent to the Emotion Detector, which calculates the feelings based on the present Facial Action Units. An in-depth explanation of this flow is given in Sections 4.2.3, 4.2.4 and 4.2.5.

CNN based Emotion Detector

As a way of comparing which State-of-the-Art methods can be more efficient, some experiments in regards to CNN based ways were also done. Several methods were tried, in order to find the most efficient one for the proposed issue, by starting with CNN models from scratch to using transfer learning on already trained models and fine tuning to increase the accuracy of the model. The implementation details are given in Sections 4.2.6 and 4.2.7, while all the experiments done are tackled in Chapter 5.

4.2.3 Facial Landmarks Detection

Currently, several facial landmark detection algorithms are implemented, all of them having some elements in common. One of the similarities would be that they all try to localize and label a few distinctive facial regions, like mouth, right and left eyebrows, right and left eyes, nose and jaw.

The chosen method for this solution is the one developed by Kazemi and Sullivan [38], which is included in the dlib open source library. They tackled the issue in determining the facial landmarks when having only one image as input, by considering the intensity of the pixels. This means that no feature extraction is taking place, therefore increasing the speed of the computations.

The method estimates the location of **68 *(x, y)* coordinates**, that map the facial structures on the face. For every run, the resulted points refer to the same facial section, making it easier in retrieving the segments and using them for further computations.

The figure 4.5 and table 4.1 present how the 68 points are divided on the face, as well as their corresponding face segments.

In the current paper, the detected landmark points are used to determine how facial segments are moving from one frame to another. This is helping to detect the specific Action Units defined by Paul Ekman.

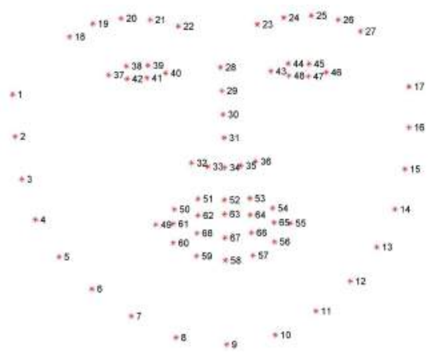


Figure 4.5: 68 points mark-up [4]

Face segment	Corresponding Landmark Points
left jaw line	1-8
chin	9
right jaw line	10-17
left eyebrow	18-22
right eyebrow	23-27
bridge of the nose	28-31
bottom of the nose	32-36
left eye	37-42
right eye	43-48
outer edge of the lips	49-60
inner edge of the lips	61-68

Table 4.1: Face segments with corresponding landmark points

4.2.4 Optical Flow Detection

The step that follows the Facial Landmark detection is the Optical Flow of the faces, both of these elements being crucial in the conventional emotion recognition method.

In Chapter 3, two of the most used methods are described, the Sparse Optical Flow, which uses Lucas-Kanade method, and the Dense Optical Flow, which is based on the Gunnar Farnebäck method. Both of them are implemented in the proposed solution, also utilising a Factory Design Pattern, as it can be seen in figure 4.6. The **ImageExtractor** class is an interface that implements several other classes, which is more detailed in section 4.3, implementing also the **OpticalFlowDense** and **OpticalFlowSparse** classes.

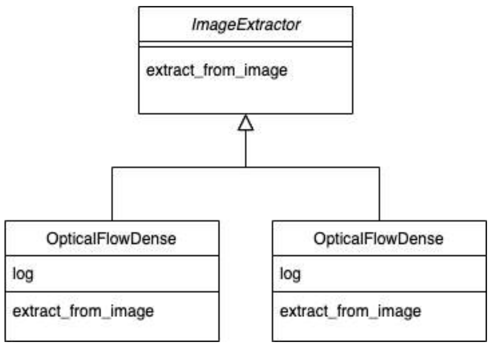


Figure 4.6: Optical Flow Classes

Even though the Sparse Optical Flow is implemented, it is mostly used for comparisons between it and the other implemented method. The information that results from applying it is deficient for the computations that follow the current step. This method also outlines how some given points have changed in regards to two frames.

Therefore, the used method for the algorithm is the Dense Optical Flow one, considering all the information that the method provides. It returns the **magnitude** and the **angle** of every point from the input.

The magnitude is normalized on the interval [0, 255], while the angle is changed considering this formula:

$$magnitude = \frac{angle * 180}{2 * \pi}$$

The values before and after they are modified can be seen in figure 4.7. For visualization, the data is stored as a HSV image, where the Hue is the modified angle, the Saturation is set by default to 255 (meaning the maximum value) and the Value is the MinMax normalization of the magnitude.

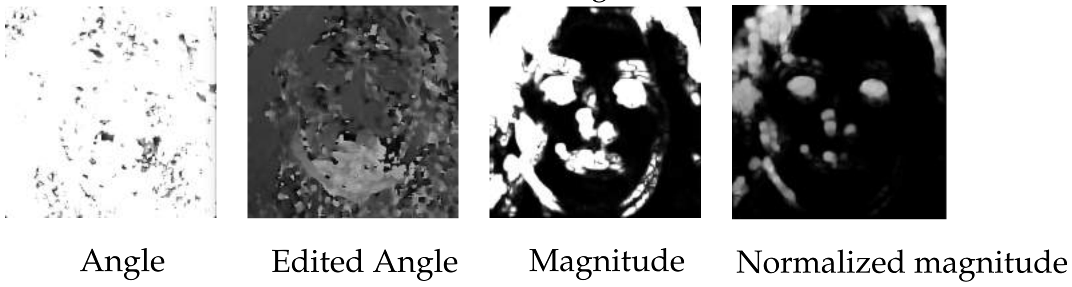


Figure 4.7: Initial and converted Angle and Magnitude

The **Hue**, in regards to an HSV image, represents the colour. The values vary from 0 to 255, where 0 is red, being followed by orange, yellow, green, blue, violet and variations between them. In regards to the Optical Flow, the Hue represents the movement of that specific pixel. For example, an upward movement leads to a colour closer to yellow - red (i.e. closer to 0), while a downward movement leads to a colour closer to blue - purple (i.e. closer to 255).

The **Value**, in an HSV image, determines the lightness of a picture. The values also are in a range from 0 to 255, where 0 means that the frame has a darker shade and values close to 255 mean that the image is full of colour. Compared to the Dense Optical Flow, the Value is proportional to the speed. Therefore, a brighter colour (i.e. one close to 255) means that the movement had a major intensity.

The **Saturation** is set by default to 255, as it does not influence the result and the only points of interest are the Hue and the Value.

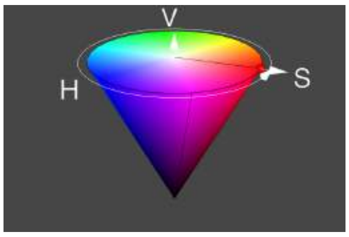


Figure 4.8: HSV Cone

Figure 4.8 presents the HSV Cone, where all the values are 255. All the before mentioned details can be observed, how the Hue or the Value vary depending on their intensity.

4.2.5 Emotion Detection

Considering the flow presented in figure 4.4, by reaching this point, the facial landmark points and the optical flow of the current frame are computed.

The Conventional Emotion Detection method is based on the work done by Paul Ekman in determining the Facial Action Units. As it is mentioned in Chapter 2, the facial musculature is divided into 46 component movements. The current solution tackles the recognition of 13 out of these 46 movements, taking into account the emotions that the algorithm is detecting, by using the results from the previous computations.

Consequently, the basic emotions and their corresponding composing Action Units are described in table 4.2.

Basic Emotion	Corresponding AUs
happiness	AU6 and AU12
sadness	AU1, AU4 and AU15
surprise	AU1, AU2, AU5 and AU26
fear	AU1, AU2, AU4, AU5, AU7, AU20 and AU26
anger	AU4, AU5, AU7 and AU23
disgust	AU9, AU15 and AU16

Table 4.2: Emotions with Corresponding AUs

It is to be mentioned the fact that in the case that an emotion is not detected, then the algorithm considers it as being neutral.

Each Action Unit is referring to a specific movement and a certain area of the face. Therefore, the landmark points are used in order to determine the segment of interest on the entire face, while the optical flow determines how those certain points have moved in consecutive frames, considering the intensity and the direction of the motion.

In the following subsection, each tackled Action Unit is described, taking into account the way a human shows it, considering Paul Ekman's work, as well as explaining how the facial landmark points and optical flow determine the motion.

Action Unit 1 - Inner Brow Raiser

The current Action Unit determines the pulling of the inner portion of the eyebrows upwards. Moreover, it produces an oblique shape to the eyebrows. Some other elements would be the skin in the center of the forehead to wrinkle horizontally, but the current method is not taking into account this detail, as it is not detected by the facial landmark points.

The proposed solution is extracting the inner corner of both eyebrows (meaning the landmark 20-22 and 23-25) and it checks whether all the landmark points have an upward movement, with a low intensity of the motion.

The reason for checking that the motion had a low intensity is because between two frames the motion cannot be that fierce. In the case that this kind of motion exists, it means that the two frames were very different and cannot determine a true value for the detected emotion.

Action Unit 2 - Outer Brow Raiser

This Action Unit is pulling the lateral (or outer) portions of the eyebrows upwards, which is a similar movement to the one described before. This motion determines an arched shape to the eyebrows, as well as the appearance of short horizontal wrinkles above the lateral portions of the eyebrows.

The algorithm is extracting the needed landmarks, meaning the outer part of both eyebrows: landmarks 18-20 and 25-27. After the extraction, it checks if all the points have an upward movement, while the intensity is also low.

Action Unit 4 - Brow Lowerer

The current Action Unit has several effects on the face. First of all, it pulls the eyebrows closer together. Furthermore, this Action Unit has different ways to be expressed depending on the circumstances. This means that sometimes only the inner portion of the eyebrow is lowered, or both inner and central parts are lowered, or the entire brow is lowered. Lastly, it pushes the eye cover fold downwards and it may narrow the eye aperture.

As in the other cases, the current method extracts the points of interest: the left (landmarks 18-22) and right (landmarks 23-27) eyebrows. Considering that only some points might change, the algorithm determines this Action Unit if at least one of the extracted points has a downward movement with low intensity.

Action Unit 5 - Upper Lid Raiser

Action Unit 5, as the name is suggesting, refers to the raise of the upper lid. This determines the eye aperture to be widen, as well as other effects such as the raise of the upper eyelid to such an extent that it disappears from the view, or that the upper portion of the eyeball is more exposed. Moreover, the shape of the upper rim of the eye is being changed, considering the fact that the medial and/or lateral portions are pulled up.

The proposed solution selects only the relevant landmarks for this Action Unit: the upper left eye (37-40) and the upper right eye (43:46). After the selection is done, the movement of all points is checked, and in order for it to be determined, every landmark needs to have an upward movement, where the intensity of the motion is also low.

Action Unit 6 - Cheek Raiser and Lid Compressor

The next Action Unit that is brought into discussion is the sixth one, which refers to the cheeks and lids. On a real life environment, the consequences of it vary from drawing the skin towards the eye from the temple and cheeks, to narrowing the eye aperture by pushing the skin surrounding the eye towards the eye socket or lowering the lateral portion of the eyebrows to a small extent.

In this case, the algorithm verifies whether all the landmark points have an upper movement with a small intensity, where the points of interest are the lower left eye (41-42) and the lower right eye (47-48).

Action Unit 7 - Lid Tightener

The current Action Unit is also referring to the lid, but this time to a tightener motion of it. As a result, the eyelids are tightened, the eye aperture is narrowed, as well as the raise of the lower lid.

The check of this Action Unit is more complex, considering the fact that the effects are to a bigger extent, and not only to a certain part of the landmarks. Therefore, it checks that some of the lower left and lower right eye parts have an upper movement. Moreover, it also determines whether some of the upper left and upper right eye points have a downward movement.

Action Unit 9 - Nose Wrinkler

Now that the movements of the eye and eyebrows have been tackled, the following Action Units describe the movement of more lower face parts, as it is in the current case, where the point interest is the nose. The wrinkling of the nose has many effects on have the face looks like, by pulling the skin along the sides of the nose upwards, towards the root of the nose. This causes wrinkles to appear along the sides of the nose and not only. It can also narrow the eye aperture and to pull the center of the upper lid upwards.

Even though AU9 has affects on many other face parts, the proposed algorithm tackles only the movement of the bottom part of the nose (meaning landmarks 32-36). It checks whether all the points have an upward motion with a slow movement.

Action Unit 12 - Lip Corner Puller

Several AUs referring to the lip movement are described in the current and the following subsection. The current one, the Lip Corner Puller, has the movement that is suggested by the name. Therefore, the corners of the lips are pulled back and upward (an oblique motion), deepening the nasolabial furrow.

The algorithm takes into account only the movement of the left and right outer parts of the lip, meaning the points 49-50 and 60-61 for the left part and the points 54-56 and 65 for the right part. To determine if the Action Unit is present, the solution checks whether some of the points have an upward movement.

Action Unit 15 - Lip Corner Depressor

The current Action Unit has an opposite action to the AU12, meaning that the corners of the lips are pulled downward. The lips are also changed, resulting in an angled down shape at the corner, the lower lip being stretched horizontally.

To check whether this AU is present or not, the same points as before are taken into consideration, but instead of verifying whether they have an upward motion, it is checked if any point at both sides has a downward movement.

Action Unit 16 - Lower Lip Depressor

Compared to AU12 and AU15, the Action Unit that is brought into discussion is referring to the whole lip being depressed, not only the corners. The movement is identified as pulling the whole lower lip down, which causes the stretch of it, being also pulled laterally. It can also result in a lips part, exposing the lower teeth.

Since the current Action Unit deals with the whole lower lip, the required landmark points are easily detected (49, 55-61 and 65-68). Moreover, the action need to be of lowering the points, also with a low intensity.

Action Unit 20 - Lip Stretcher

By pulling the lips back laterally, the mouth is elongated and the AU20 is created. Not only does it have these effects, but the lips also become flattened and stretched by the lateral pull, or the lip corners may be raised or lowered to a limited extent, the main movement being horizontal.

For the current Action Unit, the corners of the lips are again taken into account. But instead of checking whether the movement is either upward or downward, this time it is checked for an horizontal movement for the left outer lip and right outer lip.

Action Unit 23 - Lip Tightener

In a real environment, AU23 outlines how the lips are tightened and the effects on the face in regards to this action. The red part of the lips is appearing more narrow, or to roll inwards and almost disappear from the view. Some small wrinkles might also be produced in the area surrounding the lips.

The proposed solution is extracting the upper and lower parts of the lips, but without taking the corners into account, since they should stay in place. Therefore, the landmarks of interest are 50-54 for the upper part, and 56-60 for the lower part. After the detection, the algorithm checks for the upper movement of the lower lip and for the lower movement of the upper lip.

Action Unit 26 - Jaw Drop

The last Action Unit that is taken into consideration is AU26. The results of it are that the mandible is lowered by relaxation, or that the mouth appears as if the jaw has dropped or fallen with no sign of the jaw being pulled open. Moreover, the time of the action is relatively slow, as the muscle relaxes.

In this case, the chin is the only point of interest in the checks of the movement (landmarks 6-12). The algorithm verifies whether the extracted points have a lower movement, where the speed of the motion is slow.

Considering that all the Action Units that are being detected by the current solution have been detailed, the only part left in the Conventional Emotion Detector algorithm is to determine the emotion based on the present AUs. This is done by making use of a dictionary and checking for every emotion whether all the necessary Action Units are present or not, based on table 4.2. Moreover, for a face, only one emotion is allowed to be present. Lastly, in the case that no emotion is being matched, the face is said to have a neutral state.

In the following part of the paper, the Action Units that compose the emotions are visually described, to make the mentioned notions easier to be understood. Initially, the neutral state is presented in figure 4.9, such that action units can be compared to how the face is initial and how it is when the action unit is present.

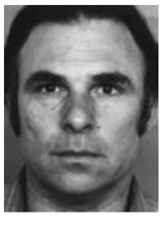


Figure 4.9: Neutral state[5]

The first emotion, which is the most basic one in regards to the way it is compounded is happiness. It is made only of AU6 and AU12, as it can be seen in figure 4.10.

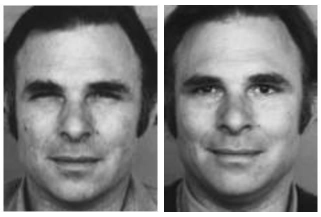


Figure 4.10: Corresponding AUs for Happy

Sadness is another emotion that is detected by the proposed solution. In regards to the required Action Units for it to be detected, AU1, AU4 and AU15 compose it. They can be observed in figure 4.11.

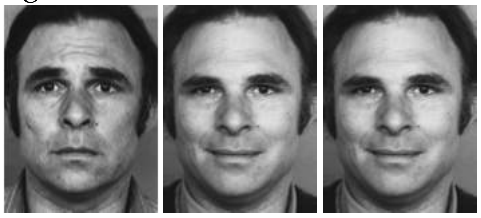


Figure 4.11: Corresponding AUs for Sad

AU1, AU2, AU5 and AU26 are the Action Units required for surprise to be detected, as it can be seen in figure 4.12.

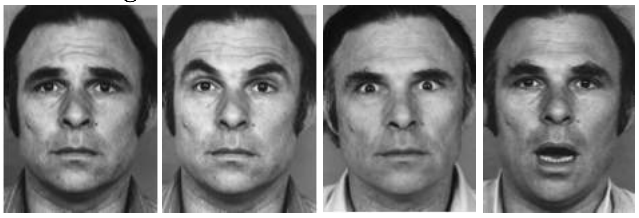


Figure 4.12: Corresponding AUs for Surprised

The most complex emotion that the proposed solution is detecting is the fear, which has in its composition seven Action Units (AU1, AU2, AU4, AU5, AU7, AU20 and AU26). All the Action Units can be seen in figure 4.13.

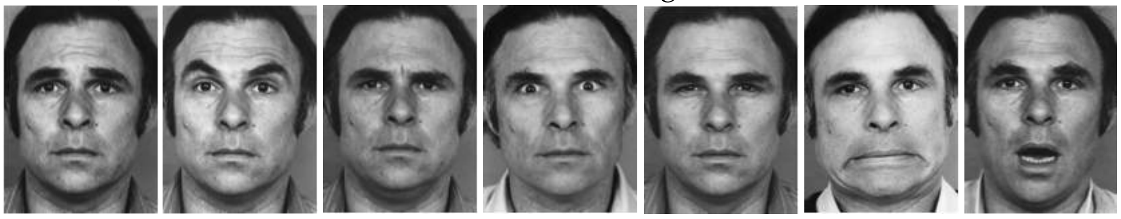
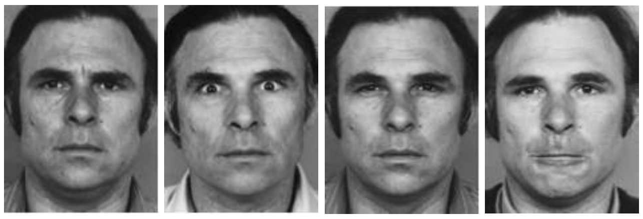


Figure 4.13: Corresponding AUs for Scared

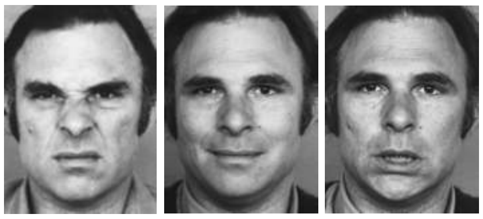
Anger is another facial emotion that the algorithm is detecting, having as corresponding Action Units AU4, AU5, AU7 and AU23. The before mentioned AUs can be observed in figure 4.14.



AU4[5] AU5[5] AU7[5] AU23[5]

Figure 4.14: Corresponding AUs for Angry

The last emotion that the proposed solution is taking into account is disgust, which is made of three AUs: AU9, AU15 and AU16, which are visually represented in figure 4.15.



AU9[5] AU15[5] AU16[5]

Figure 4.15: Corresponding AUs for Disgusted

Finally, as a way of showing how all the before mentioned elements in the Conventional Emotion Detector flow together, figure 4.16 is described. The blue bounding box that is surrounding the face, the different colours from the optical flow and how the face moved before, as well as the landmark points that are set on the crucial facial segments can be seen. Lastly, the emotion presented in the current frame is also shown on the top left side of the facial bounding box.

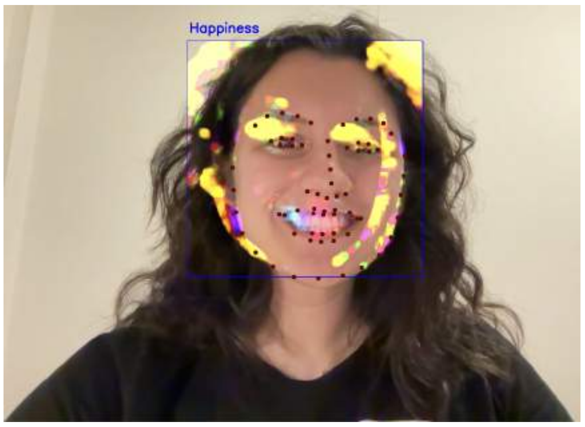


Figure 4.16: Conventional Emotion Detector complete flow

4.2.6 Data Generators

The next method that refers to Facial Emotion Recognition is a CNN based one. Considering the fact that all the details regarding the Conventional method are previously explained, the next subsections focus on the deep learning one. Several models are tried, in order to find the one that has the best accuracy and speed. But before jumping to the models explanation, one of the most crucial part in developing a deep learning model is to create Data Generator classes for the training and validation datasets.

Following the Factory Design Pattern, a custom class, **DataGenerator** is created, having several other classes inheriting from it, depending on the datasets (for example, **DataGeneratorKDEF** and **DataGeneratorJAFFE**). The specific variables, as well as the methods for each class can be seen in figure 4.17.

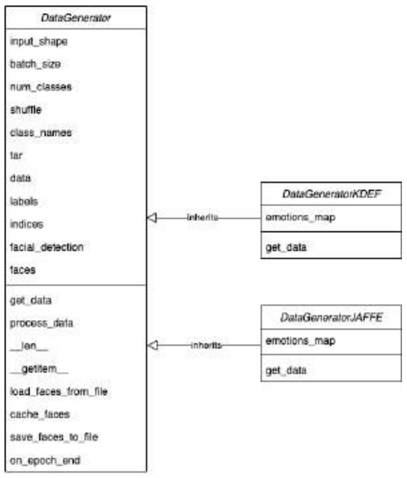


Figure 4.17: Data Generator classes

The flow and the motivation behind creating a generator is what makes the class to be so complicated, both in regards to the number of variables and number of methods. The general flow of a DataGenerator is compounded of several steps.

First of all, when the object is created, several methods are being called. As a way of speeding up the execution and not needing to load from memory each photo and its corresponding label, this information is stored in a tar file. This specific tar file is created in the moment of the first ever initialization of an object of that type.

In the case that the tar file exists, then it is loaded and all the data is saved accordingly. Otherwise, the file needs to be created. Every photo is iterated through, pre-processed and all the faces are extracted. All the results are stored in the new created file, considering the faces present in a picture and the labels.

The DataGeneratorKDEF and DataGeneratorJAFFE classes are created because these the KDEF[8] and JAFFE[39, 40] have different ways of storing the data, as well as other codings for the emotions. For example, the first dataset marks fear as *AF*, while the second one marks it as *FE*. This is the reason that these two classes only override the *get_data* method, all the other ones being general for every dataset.

Another Generator class for the AffectNet[10] database is created, but not inheriting from the base DataGenerator. The reason for this is the high memory that is supposed to hold all the images in this dataset. Therefore, the images are loaded from memory considering each batch, every time it is called by the model.

The data generators are used for training and validating a CNN model. The batch size is defined in the beginning, and for every epoch the dataset is shuffled and batches of images of that initial size are being used.

4.2.7 Convolutional Neural Networks Models

Following the Data Generators that are explained in the previous subsection, the current one is giving an insight on how they are used when creating CNN models, as well as the way the models behave and the paradigm used for training them to get the best results, which are tackled in Chapter 5.

There are two ways of creating a CNN model. One method would be to start the model from scratch, meaning that the weights are initialised randomly and they learn everything from the beginning, considering the training data. This requires a large dataset and high computational power, and it can be less efficient and less accurate. The second way is to use Transfer Learning and Fine-tuning on already existent and trained CNN models, depending on the use case. Usually the Convolutional Networks used for Transfer Learning are pretrained on a very large dataset, like ImageNet[41], which contains 1.2 million images divided into 1000 classes.

When using Transfer Learning, two scenarios can be taken into account. The first one is when the Convolutional Neural Network is fixed as the Feature Extractor. This is done by selecting a pretrained model, removing the last fully-connected layer (which is the classifier one), and treating the rest of it as a fixed feature extractor for the new dataset. A new Dense layer is added, as well as other fully-connected layers, and a new linear classifier is trained considering the new data. The second scenario is not only to use Transfer Leaning, but also to Fine-tune the weights of the pretrained network by continuing the backpropagation. Only some layers can be fine-tuned, or all the layers in the network, depending on the use case. Moreover, during Fine-tuning, the learning rate is set to a lower value than the previous one.

Considering the notions presented previously, in the following paragraphs the used networks for recognizing the facial emotions are detailed. Because of the lack of a large dataset, the proposed solution centers around already trained CNN models, like **MobileNetV2**[42], **EfficientNetB0**[43] or the Deep Neural Network **VGGFace**[44].

MobileNetV2

MobileNetV2 is a new mobile architecture, which brings an improvement in comparison to the previous mobile networks. This CNN model is based on an inverted residual structure where the shortcut connections are between the thin bottle-neck layers. Moreover, the intermediate layer is using lightwise depthwise convolutions. Finally, the non-linearities are removed from the narrow layers. This performance of this model is measured by using ImageNet classification, COCO object detection or VOC image segmentation. The figure 4.18 presents the architecture of the MobileNetV2.

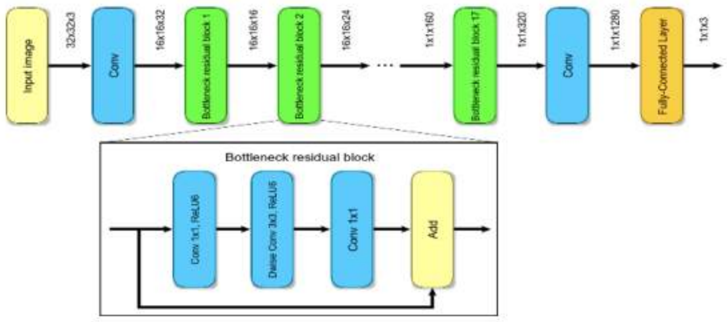


Figure 4.18: MobileNetV2 Architecture[6]

EfficientNet

Tan and Le [43] brings an innovation when developing Convolutional Neural Networks. Their work is centered around studying model scaling and identifying the fact that a better performance results from balancing network depth, width and resolution. Therefore, a family of models called **EfficientNet** is created, which obtains state-of-the-art results on ImageNet (84.3% accuracy), as well as on other datasets like CIFAR-100 or Flowers. The architecture of EfficientNetB0, which is the used model used in this solution, can be observed in figure 4.19.

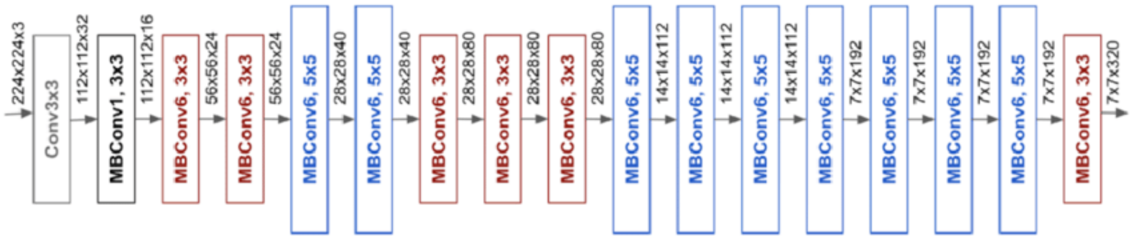


Figure 4.19: EfficientNetB0 Architecture[7]

VGGFace

The VGGFace model was developed by reasearchers at the Visual Geometry Group at Oxford and it is a state-of-the-art model. A contribution of the paper published by Parkhi et al. [44] is a description of how a very large training dataset can be developed, such that it can compete with large datasets used by Google or Facebook. The main idea is using a triplet loss embedding method, which grants state-of-the-art results on datasets like LFW and YTF. Figure 4.20 presents the architecture of the VGGFace Convolutional Network.

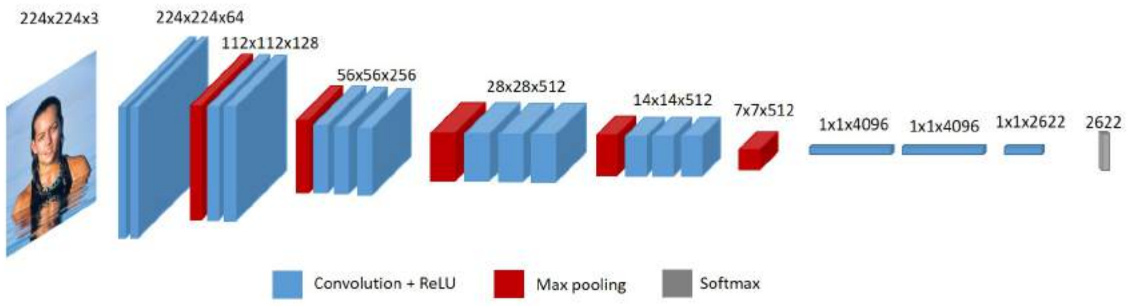


Figure 4.20: VGGFace Architecture

Every of the reminded networks is used for different trainings with several datasets. The flow of creating a model follows a similar pattern. The new models can be divided into three parts: the data processing, the feature extractor and the classifier.

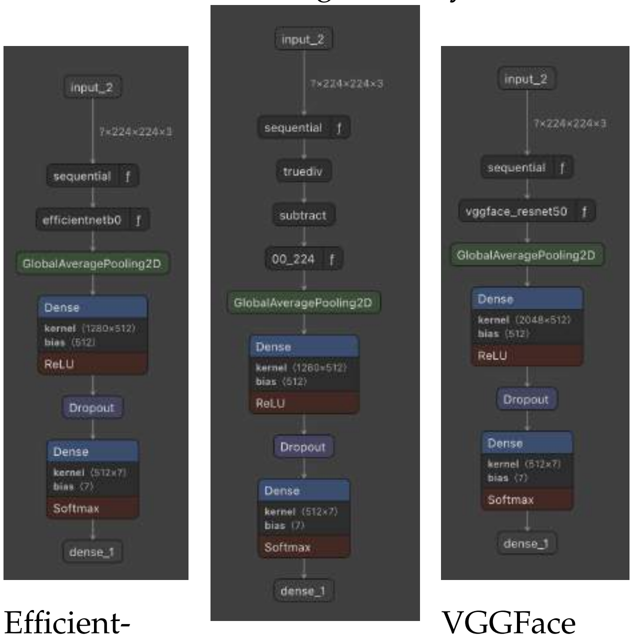
The **data processing** part of the models begins with an Input layer, where the size is given. After it, a data augmentation layer is added, which has the purpose of making the dataset more diverse, since it flips, rotates, contrasts and translates the given input. Lastly, which is an optional step required only in the case of the MobileNetV2 base model, is to add a preprocessing input layer.

The second part is the **feature extractor** part, which is done solely by the pre-trained network.

After all the features are extracted, the **classifier** is the next step. It is compounded of a GlobalAveragePooling, a Dense layer, a Dropout layer and the last layer is also a Dense layer activated by a softmax function with seven outputs, the number of classes that we classify for.

All the architectures can be observed in figure 4.21, as well as all the similarities and differences between them.

After the model is created, it is compiled by using an Adam optimizer and a SparseCategoricalCrossentropy loss. But in order for the accuracy to be maximized, to the model some fine-tuning is also applied. Therefore, a number of layers from the top of the model are set to be trainable, compiling and fitting the model once again, but with a slightly lower learning rate compared to the first time. Moreover, the number of epochs is also decreased significantly.



Efficient-NetB0 based

MobileNetV2 based

VGGFace based

Figure 4.21: Proposed CNN models

Figure 4.22 presents the result of the complete flow of running the Emotion Detector using the CNN based method. As it can be seen, the bounding box of the face is determined and the emotion shown is written on the top left part of it. This is a frame from a video capture.

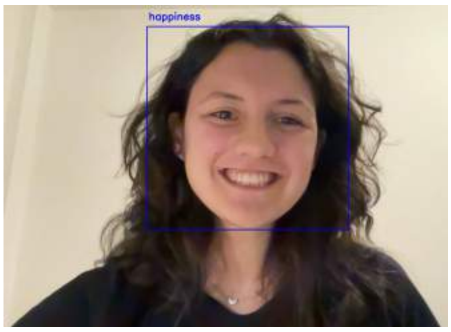


Figure 4.22: CNN based Emotion Detector complete flow

All of the three models are being trained using several datasets, and other hyperparameters being changed, reaching a conclusion that using an EfficientNetB0 model has the best accuracy for this use case, being also used as the default CNN based Facial Emotion Recognition. More details regards this matter are tackled in Chapter 5, considering the accuracy, the loss and all the trainings that are done.

4.3 Application

The purpose of this section is to present the developed application, how the Front-End is bonding with the Emotion Recognition algorithms, in regards to the functional, object and dynamic models. Moreover, all these models are detailed considering the specific diagrams, such as use case, class, sequence diagrams etc. The subsections of this chapter follow the usual steps in the development of an application.

4.3.1 Requirements elicitation

The developed application is meant for therapists, as a way of storing all their patients data, as well as getting statistics of the emotions shown by the patients during recorded sessions. Considering the use case diagram shown in figure 4.23, the functional requirements of the web application can be summed up as: login, signup, CRUD operations on patients (listing all patients, adding, updating and deleting) and getting statistics from videos.

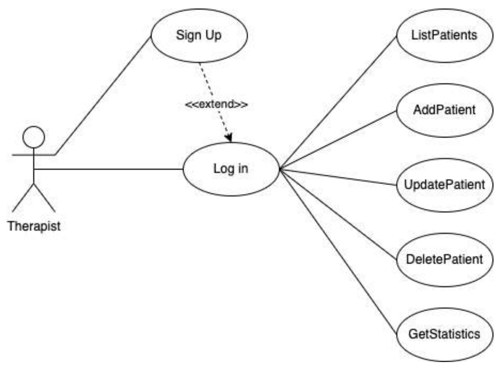


Figure 4.23: Use Case diagram

Several nonfunctional requirements have been defined, in order to provide a better User Experience to the therapists that will make use of this web application.

Considering the Usability, the application is straight forward, with buttons that redirect the user to the suggested pages. Moreover, a pastel colour palette has been chosen, as these colours have a calming effect on the user.

Moreover, the application is providing an answer to the user's request as fast as possible. The reaction time depends on the operation, as the login and the CRUD operations are done in an instant, while the statistics of the video depend on the length of the video (the time is around 4.5 frames/ second).

Lastly, the application is also tackling security issues, as non logged in or non signed up people cannot access it, also the user has access only to the specific objects. Moreover, the login in token expires after a set amount of time, also as a secure way of keeping the data as private as possible, on top of storing all the sensitive data in a hashed way.

4.3.2 Requiremenets Analysis

The second stage in developing an application is the analysis part, which focuses on producing a model of the system discussed about during Requirements elicitation. Figure 4.24 shows how all the classes of the backend of the application interact with each other. It represents the object model of the software product.

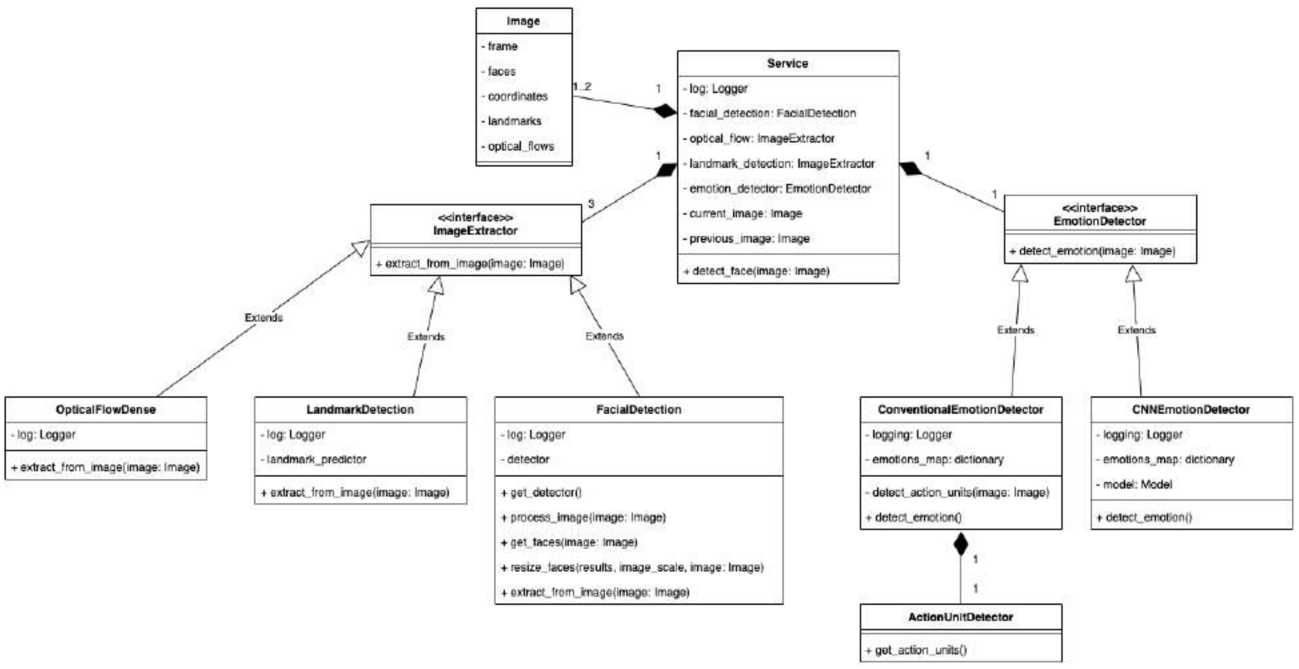


Image
- frame
- faces
- coordinates
- landmarks
- optical_flows
1..2
1
Service
- log: Logger
- facial_detection: FacialDetection
- optical_flow: ImageExtractor
- landmark_detection: ImageExtractor
- emotion_detector: EmotionDetector
- current_image: Image
- previous_image: Image
+ detect_face(image: Image)
1
3
<<interface>> ImageExtractor
+ extract_from_image(image: Image)
1
1
<<interface>> EmotionDetector
+ detect_emotion(image: Image)
Extends
Extends
Extends
Extends
Extends
OpticalFlowDense
- log: Logger
+ extract_from_image(image: Image)
LandmarkDetection
- log: Logger
- landmark_predictor
+ extract_from_image(image: Image)
FacialDetection
- log: Logger
- detector
+ get_detector()
+ process_image(image: Image)
+ get_faces(image: Image)
+ resize_faces(results, image_scale, image: Image)
+ extract_from_image(image: Image)
ConventionalEmotionDetector
- logging: Logger
- emotions_map: dictionary
- detect_action_units(image: Image)
+ detect_emotion()
CNNEmotionDetector
- logging: Logger
- emotions_map: dictionary
- model: Model
+ detect_emotion()
1
1
ActionUnitDetector
+ get_action_units()

Figure 4.24: Class diagram

It can be observed that the Service is the class around which all the objects are created, is the class that leads all the operations after the request is received from the front end. It has in its compound all the needed classes to detect the emotions, such as FacialDetection, OpticalFlowDense, LandmarkDetection, as well as EmotionDetector. All the classes are based on using the Image class, which holds for each image all the needed details.

As in the module explanations that are done in Section 4.2, the Factory Design Pattern is used in the development of the application. The ImageExtractor class is the base class for every module that deals with extracting data from an initial frame. Moreover, this can also be observed in the EmotionDetector interface, that is the base class of both the Conventional and CNN algorithms. In both cases the base class has one abstract method that determines the usage of the class (*extract_from_image* for ImageExtractor and *detect_emotion* for EmotionDetector).

This class diagram is not showing all the developed modules, as the FacialDetection class is extended by three other classes, as it is explained in Subsection 4.2.1. Moreover, some of the developed classes mentioned previously are used only for the CNN model creation and are not related to the Service module. The CNNEmotionDetector class loads the model, making use of it. Furthermore, some other modules that are further explained deal with the endpoints, or with all the CRUD operations done for users / therapists by storing the data to a database.

While the class diagram represents the object model of the software product, the sequence diagram is one of the dynamic models, which focuses on the behaviour of the system. Figure 4.25 represents the sequence diagram of an user/ therapist that is not signed up and wants to use the application to get statistics in regards to the emotions shown in a video.

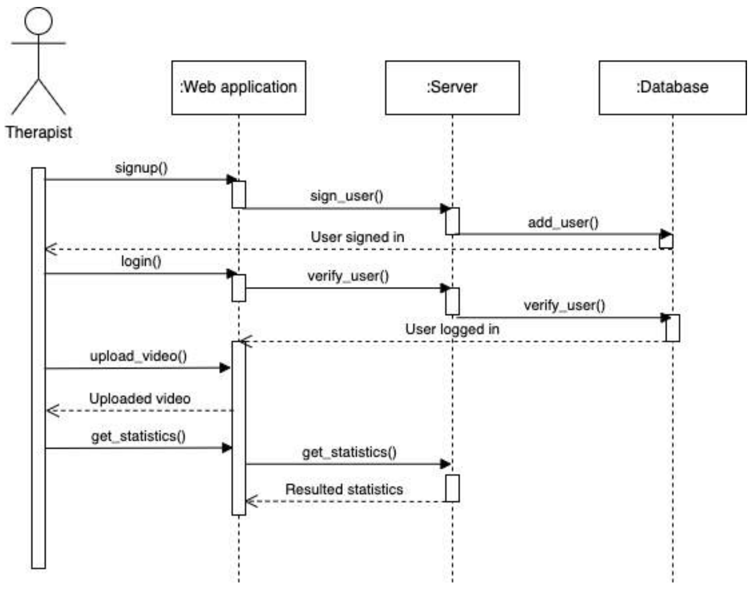


Figure 4.25: Sequence diagram of get statistics

The user only interacts with the front end part of the application, where the requests are sent to the server. If it is the case, the server communicates with the database to store the data.

The flow starts with the user getting on the sign up page. After filling all the details, they are sent to the server and stored in the database. Since now the user data is stored, they can log in using the credentials. Once again a request is made to the server, which check in the database if the username and the password correspond to an existent user. The current diagram shows the happy scenario, where the user has given the right data and it is logged in.

Now that it is logged in, they can access the statistics page. The first step is to upload a video, which is done only on the front end part and it is not sent to the server to be stored. After all the required parameters are filled in by the user and the GetStatistics button is clicked, the data is sent to the server. It calls the user preferred Emotion Detector and after all the computations are done, the results are sent back and shown on the front end. With the last step the flow ends, allowing the user to use other functionalities.

4.3.3 System design

Considering all the functional and non-functional requirements presented during the elicitation phase, as well as the classes and how the flow of the application should be, the whole system is also designed. Figure 4.26 presents the elements of the system. Therefore, from the PC, the user is interacting with the Ionic React developed web application, that is sending HTTP requests through the endpoints determined by the FastAPI server, having all the data stored in a PostgreSQL database.

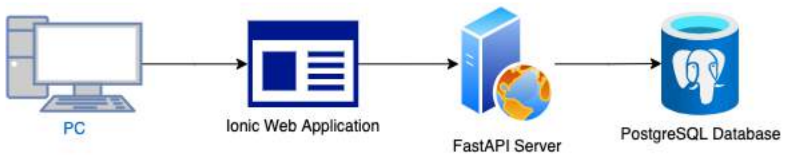


Figure 4.26: Full system design

In the following paragraphs, the three components of the system are detailed, following the user flow, from the Web application, to the Server and, finally, the Database.

Ionic Web Application

The web application has been developed using the Ionic React framework, considering the easiness of migrating to mobile applications from an one of this kind. The application has in its compound several pages, allowing the user to easily navigate through them.

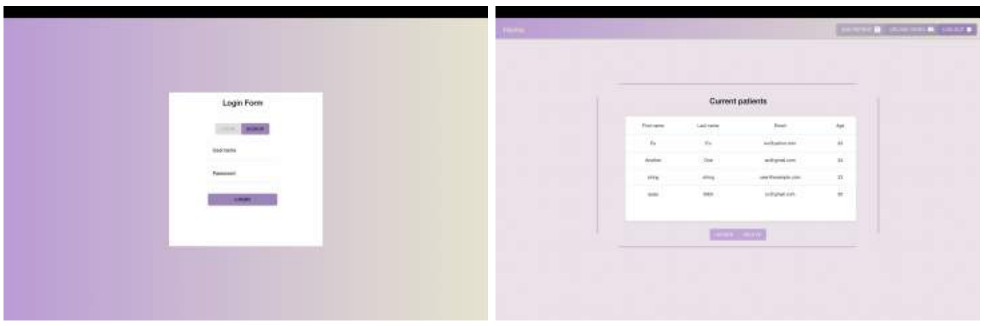


Figure 4.27: Login and Home pages

When running the application, the first page that is shown is either the Login or the Home page, depending whether the user has been already logged in the last 30 minutes or not. These two pages can be observed in figure 4.27. The Login page can redirect the user to the Signup page in the case that they are not already registered as users. Moreover, after the user logs in, it is redirected to the Home page.

The Home page presents, as the main information, all the saved patients for the logged in user, as a list. In the navigation bar of the page, a few button can be observed. The Home button redirects the user to the Home page when clicked, the Save patient button shows the corresponding page, as well as the Get statistics button. The Sign out button deletes the stored token in the current storage and logs out the current user. Moreover, when clicking on any patient, two buttons appear, to either delete or update that patient.

The Add/Update page, that can be remarked in the left side of figure 4.28, lets the user, as it is suggested by the name, either add or update a patient. In the case that the Save patient is pressed, the inputs are empty, but if the Update button from the Home page is pressed, the inputs are loaded with the current values.

Figure 4.28 also presents the Statistics page. Here, the user can upload a video, decide the preferred method for the detector, set the name of the Patient if it exists. Optionally, the user can set how many frames to be taken into account when computing the statistics. If that field is left empty, then all the frames are taken into account. After all the computations are done, the right side graph is filled with the current data to be seen by the therapist.

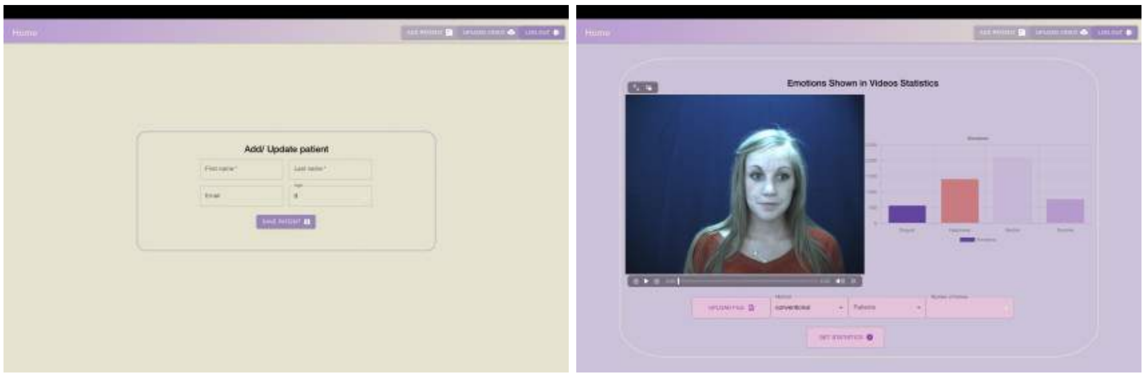


Figure 4.28: Add/Update and Statistics pages

FastAPI Server

The Server is developed using FastAPI, which is a web framework for building APIs with Python. The choice of using it was done considering the fact that the backend is developed using that programming language. Considering that the backend has been detailed previously, this Subsection is considering only the endpoints through which the front end part is communicating to the back end one.

All of the endpoints, by presenting the links, the type of the request and the headers content-type are presented in table 4.3.

Link	Type	Headers Content-Type
/signup	POST	application/json
/login	POST	application/json
/patient	GET	N/A
/patient	POST	application/json
/patient/{patient_id}	PUT	application/json
/patient/{patient_id}	DELETE	application/json
/patient/upload	GET	multipart/form-data

Table 4.3: Endpoints

There are endpoints that deal with the authentication or registration of the user, of the type POST. Some other endpoints manage all the CRUD operations on patients, using the specific types for a request of that way. A special endpoint is the one dealing with the statistics part of the application, as it required the sending of a video through an HTTP request. The Headers Content-Type is multipart/form-data, as the video and the other data is being sent using FormData.

PostgreSQL Database

The last part of the developed application is the PostgreSQL database. Here all the necessary data is stored. For now, the database is compounded of only two tables, the User (therapist) and the Patient. They are in a many to many relationship, where the User can have multiple Patients, but a Patient is not bounded to only one Therapist. the tables and the relationship between can be observed in figure 4.29.

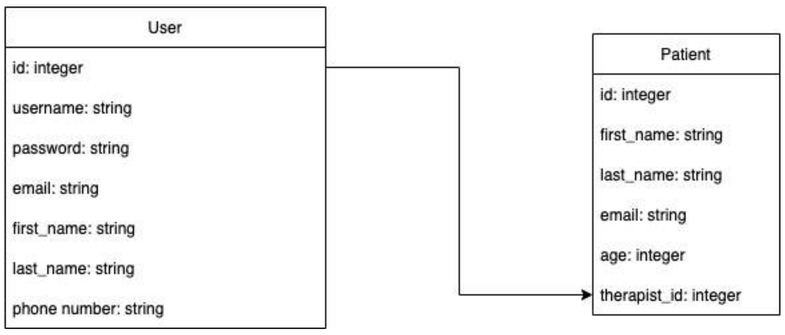


Figure 4.29: Database design

Chapter 5

Experimental results

Developing the models described in Section 4.2.7, several experiments are required to be done, in order to decide for the best model of the current use case. This chapter is tackling the datasets used to train and validate the networks, as well as presenting an overview on the how the Transfer Learning and the Fine-tune for each CNN behaved, and comparing the methods with other already existent ones.

5.1 Datasets

In regards to the datasets, the proposed solution makes use of datasets composed only of faces, whether they are photos or videos. The datasets based on photos are utilised for the Convolutional Neural Networks, while the video dataset that is detailed in the following subsections is used for testing the way the Web application is working, since it is not specialised on emotions recognition.

5.1.1 KDEF

Lundqvist et al. [8] introduced the Karolinska Directed Emotional Faces (KDEF) dataset in 1998. It is a database of pictorial emotional facial expressions compounded of 4900 pictures. 70 individuals (35 females and 35 men) with the age between 20 and 30 display 7 different emotional expressions (fear, anger, disgust, happiness, neutral, sadness and surprise) from 5 different angles. Moreover, the actors were supposed not to have any accessories, to have a beard or to wear any make-up. The subjects were instructed to try to evoke the emotion that was to be expressed and to make the expression strong an clear, while looking as natural as possible. Two of the actors of the KDEF dataset can be observed in figure 5.1

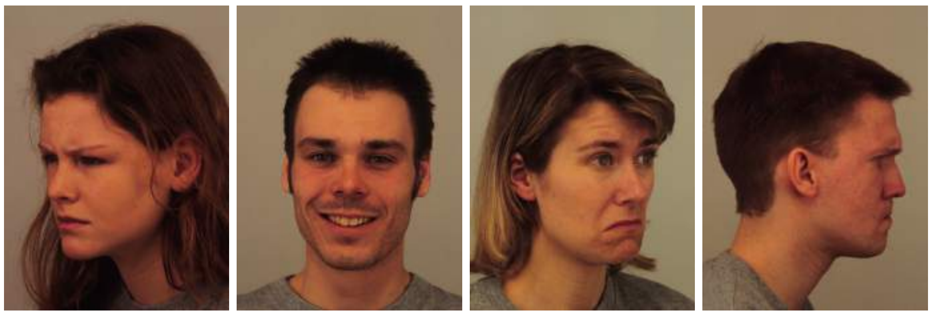


Figure 5.1: Images from KDEF dataset[8]

5.1.2 JAFFE

The JAFFE dataset is introduced by Lyons et al. [9], database consisting of 219 facial expression images. Ten subjects display 3 or 4 examples of each of the six basic facial expressions, as well as the neutral face. The expressors consist only of Japanese females, which means that this dataset can be integrated with other datasets to make the input data more diverse and to make the recognition more reliable. Each subject has the hair tied away from the face to expose all the important zones of the face. Lastly, the images are printed in monochrome, as it can be seen in figure 5.2.

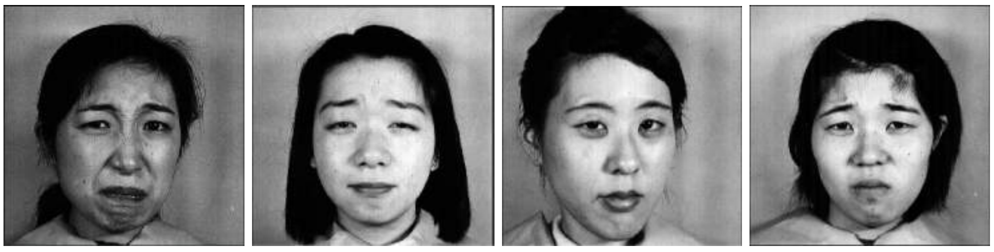


Figure 5.2: Images from JAFFE dataset[9]

5.1.3 AffectNet

The AffectNet[10] database contains about 1 million facial images collected from the internet. This is done by querying three major search engines by using 1250 emotion related keywords. Out of those initial pictures, approximately 420k are manually annotated for the seven discrete facial expressions. But those pictures do not show only people, there are some that are marked as uncertain, where there is no expression or where there are no faces. Moreover, even for the expressions, the data is divided unevenly, such that there are around 4300 images for disgust, while the pictures showing happiness are around 134k. Because of this reason, during training, the dataset is balanced, such that there is an equal number of pictures for every emotion. Furthermore, the images that do not show any of the seven emotions that this paper is tackling are being removed.

Ever image in the database has a fixed size of 224 x 224 and the faces are cropped, so there is no need for pre-processing the data before using it for the trainings. Some images from this dataset can be observed in figure 5.3.

Figure 5.3: Images from AffectNet dataset[10]

5.1.4 DISFA

The Denver Intensity of Spontaneous Facial Action Database (DISFA)[11] is a non-posed facial expression database used for automatic action unit detection. The database contains videos of 27 adult subjects (out of which 12 females and 15 males), with different ethnicities (Asians, Euro-Americans, Hispanics and one African-American person) and ages varying from 18 to 50 years. Each video is 242 seconds long, each of the subjects being alone when the video is taken. Even though this database is specialised for action units and landmark points detection, the data can be used for testing the web application, since the subjects show different emotions during the video. Some frames can be observed in figure 5.4.

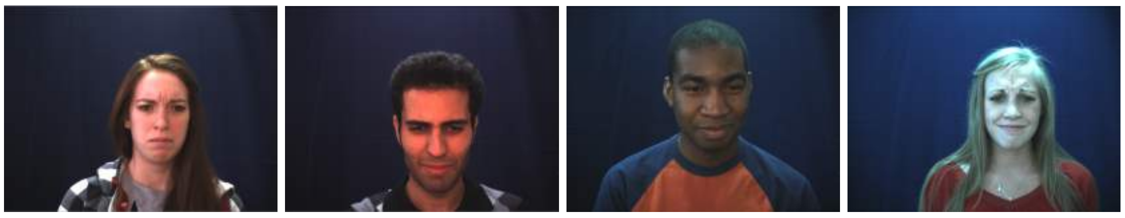


Figure 5.4: Frames from DISFA dataset[11]

5.2 Model training

Using the mentioned datasets and the base CNN models detailed in Section 4.2.7, several models are trained using Transfer Learning and Fine Tuning with the purpose of recognising facial emotions. Table 5.1 outlines all the completed trainings, each of the modified values when running a training, as well as the results both for the Transfer Learning (TL) and Fine Tuning (FT) if present.

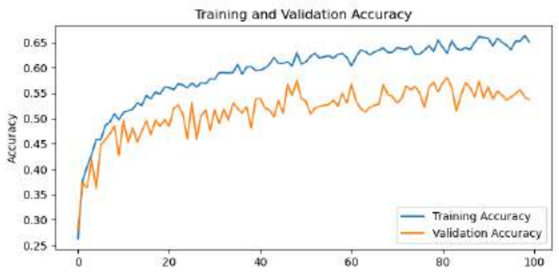
Id	Base Model	Epochs	Dataset	Batch size	Learning rate	Results TL	Results FT
2	EfficientNetB0	100 + 25 FT	KDEF	32	0.005 & 0.0005	acc. 50%	acc. 83%
3	EfficientNetB0	50 + 25 FT	AffectNet	32	0.001 & 0.0001	acc. 39%	acc. 48%
4	MobileNetV2	50 + 25 FT	KDEF	64	0.01 & 0.001	acc. 38%	acc. 39%
5	MobileNetV2	50 + 25 FT	KDEF	32	0.01 & 0.001	acc. 44%	acc. 46%
6	MobileNetV2	75 + 25 FT	JAFFE	16	0.01 & 0.001	acc. 55%	acc. 59%
7	MobileNetV2	75 + 25 FT	JAFFE	16	0.005 & 0.0005	acc. 59%	acc. 59%
7	MobileNetV2	40 + 15 FT	AffectNet	64	0.0005 & 0.00005	acc. 40%	acc. 41%
8	VGGFace	75 + 25 FT	KDEF	32	0.01 & 0.001	acc. 57%	acc. 63%
9	VGGFace	75 + 25 FT	JAFFE	16	0.01 & 0.001	acc. 73%	acc. 86%
9	VGGFace	30 + 10 FT	AffectNet	16	0.001 & 0.0001	acc. 50%	acc. 51%

Table 5.1: Models with corresponding trainings

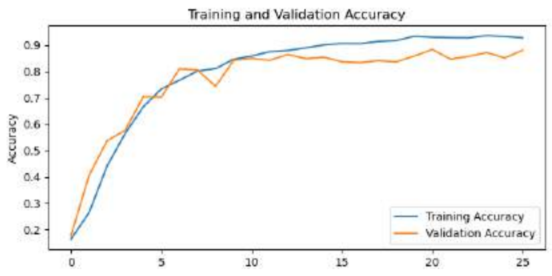
In order to reach the best results, there are several combinations tried considering the base model, the number of epochs, the dataset used, as well as the batch size and the learning rate. Each of the base model is trained using the three mentioned datasets (KDEF, AffectNet and JAFFE), where the batch size is depending on the used database. As for the epochs, the prefered number is around 75, with a few exceptions, for the Transfer Learning, while the Fine Tuning is done with a fixed number of 25. Lastly, the learning rate also varies for the Transfer Learning, with it being either 0.01 or 0.005, the learning rate for the Fine Tuning between 1/10 of the initial one. Some parameters used for training AffectNet based models differ, as it is observed a tendency of the values to stagnate after a certain number of the epochs, considering a certain loss.

It is observable that using Fine-Tuning leads to better results also in accuracy, as well as for the loss. Therefore, only the models that also used Fine Tuning are being taken into account.

Considering that these trainings are on different datasets, the network that is used by the application to detect the facial emotions is the EfficientNetB0 based one. The reason behind this is the diversity of the KDEF dataset, while the JAFFE one presents only Japanese women, as well as the limited number of data in the second database. Moreover, the AffectNet database seems harder to train, considering the input data. Wen et al. [45] scored the best accuracy for training a model with only this dataset, of 65.69%, while the best accuracy obtained by the current paper is 51%. Figure 5.5 presents the full trainings of the chosen EfficientNetB0 network.



Training and Validation accuracy during Transfer Learning



Training and Validation accuracy during Fine Tuning

Figure 5.5: Graphics of EfficientNetB0 training

In regards to the metrics, the most important one is the accuracy of the prediction, but precision, recall and the F1-score are also taken into account, as it can be observed in Classification Report Table 5.2. Moreover, all the results from the table are mentioned in the Confusion Matrix from figure 5.6.

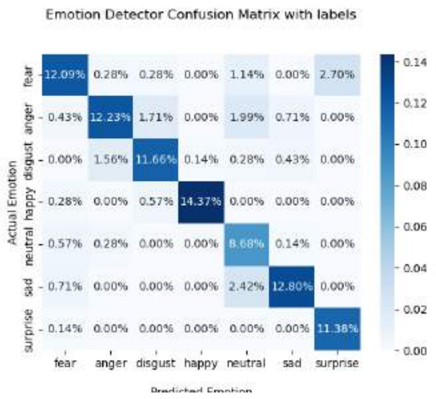


Figure 5.6: Confusion Matrix EfficientNetB0 model

Emotion	Precision	Recall	F1-score
fear	85%	73%	79%
anger	85%	72%	78%
disgust	82%	83%	82%
happy	99%	94%	97%
neutral	60%	90%	72%
sad	91%	80%	85%
surprise	81%	99%	89%

Table 5.2: Classification Report

The **accuracy** parameter gives the overall accuracy of the network, meaning the percentage of the correctly classified samples by the classifier. The current model has an accuracy of 83% after Fine-Tuning, when the initial training has only an accuracy of 50%. The formula of the accuracy is:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

, where TP - True Positive, TN - True Negative, FP - False Positive and FN - False Negative.

Another useful metric in when evaluating a CNN model is the **precision**, which tells the percentage of how many of the positive classes are truly positive. Therefore, the formula is:

$$Precision = \frac{TP}{TP + FP}$$

In the current case, besides the values presented in the following figures, the weighted precision is calculated, which has the value 85%.

Recall is another useful metric for testing the performance of a network. It is also called the True Positive Rate, as it determines how many of the positive classes are correctly predicted as positive. Taking this into account, the formula of recall is:

$$Recall = \frac{TP}{TP + FN}$$

For the EfficientNetB0 model, the weighted recall is 83%, meaning that, on average, only that percentage of the predicted classes are correctly determined to be of that specific class.

The last metric that is taken into consideration is the **F1-score**, which is the parameter that combines both the precision and the recall into a single one, by calculating mathematically the harmonic mean of these two values. Therefore, the formula is:

$$F_1 score = \frac{2TP}{2TP + FP + FN}$$

The weighted F_1 score is the same as the recall, of 83%.

5.3 Detectors comparison

This section presents different comparisons for the two proposed methods. Initially, an in-depth analogy is presented between the Conventional way of detecting emotions and the CNN one. After, it tackles how the CNN model can be compared to already existent networks.

5.3.1 Proposed detectors comparison

This paper proposes two methods of recognizing the facial emotions, one Conventional method, which is based on Paul Ekman's Action Units, as well as Optical Flow and Facial Landmark Points and a CNN method, the EfficientNetB0 based one.

In regards to the the comparison between these methods, the videos used for running the algorithms are from the DISFA dataset, choosing four videos representing two males (one Euro-American and one African-American) and two females (one Euro-American and one Hispanic). The reason for choosing this distribution is to check the differences in recognizing the feelings when having as input faces from different races or genders. Moreover, the comparison is taking into account the processing time, as well as the distribution of the emotions. In a way, this comparison can also be expanded as a comparison between emotion recognition from videos and photos, as the conventional method takes into account the previous frame, while the CNN based way considers only the current one.

In regards to the computation time, for videos having the length 242 seconds, where 4845 frames are extracted, the average is 18 minutes and 20 seconds for video, regardless of the used method. Moreover, this time is also dependent on the used device, as well as the other computations done in parallel unrelated to the application. The complete results can be observed in table 5.3. This leads to around 0.227 seconds spent / frame, which is around 4.5 frames / second the average time.

Video	Method	Time
SN001	Conventional	18m 24s 53ms
	CNN	18m 20s 43ms
SN013	Conventional	18m 8s 70ms
	CNN	18m 9s 25ms
SN018	Conventional	17m 59s 27ms
	CNN	18m 21s 5ms
SN023	Conventional	18m 46s 8ms
	CNN	18m 16s 82ms

Table 5.3: Time taken for each method

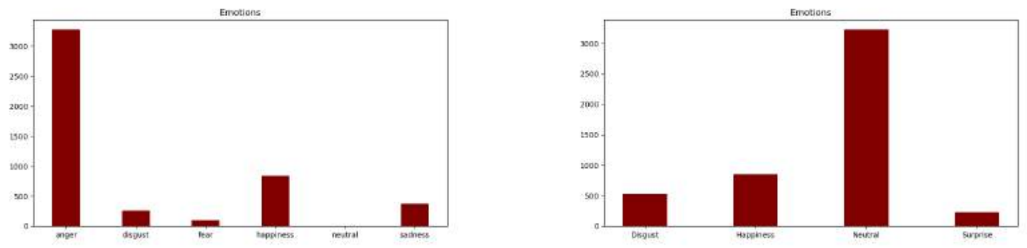
When discussing the results from each detector, several observations can be done. As a general take, it is remarked the fact that both the detectors tend to detect a specific set of emotions. This means that the Conventional method is prone to detect mostly the neutral state, while the for the CNN based one, angry is the state most determined, with some exception.

Moreover, the Conventional method seems to determine a limited amount of emotions (Disgust, Happiness, Neutral and Surprise), which are also the least complex feelings that the algorithm can detect. The EfficientNetB0 way detects a wider range of emotions, where the maximum number of types for the executed tests is 6.

Even though the emotions seem to be more accurately detected using the CNN based algorithm, they are distributed uneven, with one emotion being dominant (in most of the cases angry). As for the Conventional method, the emotions have an uniform distribution. This observation also depends on the emotions shown during the videos, so the uneven distribution during the first one can be accurate.

It is also remarked the fact that both the methods might have some errors in computation, as the Conventional method sometimes detects neutral state for a visible emotion. This is because that specific emotion might be held for longer than two frames and the detector determines it as neutral, as the actor does not move. The CNN base way also can erroneously detect some emotions, as it can mistake anger for neutral.

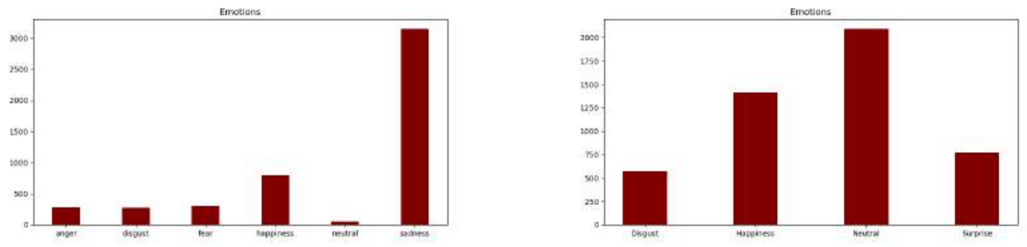
All the before mentioned observations for the comparison between the two proposed methods are presented in Figures 5.7 and 5.8, which show the emotion distribution from SN023 (the Hispanic woman) and from SN018 (the Caucasian woman). The reason for choosing these two examples is because the graphics for the other two videos are similar to the ones for SN023.



CNN method | Conventional method

Figure 5.7: Emotion distribution for SN023

The SN023 video presents a person over 242 seconds showing different emotional states. Even though the two graphs have different shapes, it is observable the fact that the happiness state is approximately the same for both of the methods ($\approx$ 1000 frames out of the 4845 ones). Moreover, the disgust state is in a similar situation, having around 500 frames detected also by the Conventional way, as well as the CNN method. The anger from the EfficientNetB0 method and the neutral state from the Conventional one have a similar distribution, which can lead to one of the before mentioned observations.



CNN method | Conventional method

Figure 5.8: Emotion distribution for SN018

In a similar way, for the SN018 video there are a few similarities and a few differences that can be observed between the two methods. Starting with the similarities, there exist some emotions that have a similar distributions over the entire video, like disgust, where the disgust state is detected in around 500 frames. The happiness state, although with a larger error margin, is being observed in approximately

1000 frames by the both detectors.

One of the differences, which can be an exception for one of the observations, is the fact that this time, the CNN based method determined sadness as being the dominant emotion, while the neutral state stayed constant for the Conventional one. Moreover, the first method shows a broader range of emotions (anger, disgust, fear, happiness, neutral and sadness), while the second one presents only four emotional states (disgust, happiness, neutral and surprise).

5.3.2 Comparison with state of the art

Chapter 2 presents in a detailed manner all the work that has been done in the Facial Emotion Recognition domain. This subsection compares some of the already defined methods to the proposed CNN based model, mainly by considering the accuracy of the networks.

For presenting the comparison, only models that have used the same dataset as the proposed EfficientNetB0 model (KDEF) are taken into consideration. The considered networks, as well as their accuracy and how they differentiate the proposed solution can be seen in table 5.4.

Work [Ref.], Year	Accuracy
Hossain et al. [30], 2021	80.92%
Proposed solution	83%
Akhand et al. [29], 2021	98.78%

Table 5.4: Accuracy comparison of proposed solution with existing works on KDEF

Both Hossain et al. [30] and Akhand et al. [29] present deep learning-based methods, as well as taking into account the full KDEF dataset. Recently only a handful of papers use all the images from this database, usually only the frontal faces are being utilised. Images with only frontal views are easier to classify compared to models that are trained using both frontal and side views.

As it is observed in table 5.4, the first deep learning-based method[30] reaches an accuracy of 82.63% on the KDEF dataset. This value is determined also by fusing the classification scores after the recognition is done by the Fine-Tuned model.

As for the pre-trained deep CNN model proposed by Akhand et al. [29], the accuracy of the network gets to 98.78%, in the case of splitting the initial dataset as 90% training data and 10% validation data.

Since the proposed detection method reaches an accuracy of 83% for the KDEF dataset, while using a the shallow CNN EfficientNetB0 method, it is situated between the two before mentioned works.

5.4 Final results

After the mentioned tests for both of the proposed Facial Emotion Recognition methods, several conclusions can result.

First of all, it is seen that the Conventional way is better at determining the neutral state, in comparison to the other method, which has determined around 100 frames in total for all the four tests that are run. The fact that the CNN based model does not determine well the neutral state can also be observed from the Table 5.2, as the precision of this emotional state is only 60%.

Secondly, the EfficientNetB0 method determines a broader range of emotions, even with smaller distribution in regards to the number of frames from each video, while the Conventional method seems to detect mostly four of the least complicated emotions. By least complicated emotions, it is meant the neutral, the happy, the surprised and the disgusted states, which are composed only of maximum four Action Units.

Finally, as the results can also be subjective in regards to the user, it is recommended that both of the methods to be run and to be taken into account the results, as it could lead to a better interpretation of the presented facial emotions. This is because the neutral face of a person can be shown as an angry face to another person.

Chapter 6

Conclusions

In conclusion, the current paper answered the objectives that were set in the beginning. Two Facial Emotion Detection methods were developed, one Conventional which was based on the work done by Paul Ekman and a CNN based one, which used Transfer Learning and Fine Tuning from the pre-trained EfficientNetB0 model.

Two types of comparisons were done in regards to the detectors. The first one was for the training of the Deep-Learning based detector, which resulted in using the before mentioned model for detection. The second one was for comparing the two developed methods, which had as results several observations regarding both ways of detecting the facial emotions. Moreover, these tests also led to the average time taken by the detectors to process a video, determined as 4.5 frames/ second.

The proposed solution also tackled the integration of the developed facial emotion recognition algorithms into a web application. The web application gave the opportunity to therapists to centralise all the patients that have sessions with them, as well as editing their data. Moreover, an interactive page for getting the statistics was developed, where the psychologist could choose several options for running the methods.

The security matter of the application was also tackled, as it allows the access to the pages only to logged in users. Moreover, all the private data like password was saved using hashing algorithms to protect against malicious attacks.

The developed methods reached promising results, that can still be improved through future developments. The algorithms performed well when they were tested on a live environment, using the webcam, but using videos from a dataset as test input led to different results. The distribution of the emotions was varying between the two methods, which outlined how the static and dynamic emotion recognition can differ in results. But even with these differences, some emotional states like happiness, disgust had a similar distribution using both these methods. Moreover, a tendency of the Conventional method detecting the neutral state and the CNN model to detect the angry state were observed. Lastly, the overall accuracy of the CNN based method could be improved, as it did not reach state-of-the-art values.

Various future work can be done to improve the proposed solution. The first improvement can be in regards to the dataset used for training the CNN model. A more varied with a better class distribution (in regards to the diversity of the people, as well as the number of images per class) could be used, such that the training is done on a broader input. Also in regards to the CNN model, it could be developed into a RNN + CNN model. In this way, this detection method would take into account the whole video and not only frames, it would be a more dynamic way of determining the emotions.

Future tests could be performed on the Conventional detection method, as a way of determining the reason for the poor distribution of classes. Moreover, the current solution was taking into account only the specific landmark point when calculating the Action Units. A further improvement would be to use a neighbourhood of the points, which could lead to better results in detection. Moreover, more Action Units could be detected in the future, such that the range of emotions is broader, leading to variations of the seven basic feelings.

Lastly, as a future development, the web application could be developed to have more functionalities also in regards to the therapist, like editing their data, having more options when authenticating etc.

Bibliography

[1] CS231n: Convolutional Neural Networks for Visual Recognition. http://cs231n.stanford.edu. 2022.

[2] Understanding of Convolutional Neural Network (CNN) — Deep Learning. https://medium.com/@raghavprabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148.

[3] Optical Flow OpenCV. https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html.

[4] Christos Sagonas, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 397–403, 2013.

[5] Paul Ekman and Wallace V Friesen. Facial action coding system. *Environmental Psychology & Nonverbal Behavior*, 1978.

[6] Ulzhalgas Seidaliyeva, Daryn Akhmetov, Lyazzat Ilipbayeva, and Eric T Matson. Real-time and accurate drone detection in a video with a static background. *Sensors*, 20(14):3856, 2020.

[7] Francis Jesmar Montalbo and Alvin Alon. Empirical analysis of a fine-tuned deep convolutional model in classifying and detecting malaria parasites from blood smears. *KSII Transactions on Internet and Information Systems*, 15:147–165, 01 2021. doi: 10.3837/tiis.2021.01.009.

[8] Daniel Lundqvist, Anders Flykt, and Arne Öhman. Karolinska directed emotional faces. *Cognition and Emotion*, 1998.

[9] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba. Coding facial expressions with gabor wavelets. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 200–205, 1998. doi: 10.1109/AFGR.1998.670949.

[10] A. Mollahosseini, B. Hasani, and M. H. Mahoor. Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, PP(99):1–1, 2017.

[11] S Mohammad Mavadati, Mohammad H Mahoor, Kevin Bartlett, Philip Trinh, and Jeffrey F Cohn. Disfa: A spontaneous facial action intensity database. *IEEE Transactions on Affective Computing*, 4(2):151–160, 2013.

[12] P Ekman. *Emotions Revealed: Understanding Faces and Feelings: Phoenix*. 2004.

[13] Paul Ekman and Wallace V Friesen. Constants across cultures in the face and emotion. *Journal of personality and social psychology*, 17(2):124, 1971.

[14] Jeffrey F Cohn, Zara Ambadar, and Paul Ekman. Observer-based measurement of facial expression with the facial action coding system. *The handbook of emotion elicitation and assessment*, 1(3):203–221, 2007.

[15] Charles Darwin. *The expression of the emotions in man and animals*. University of Chicago press, 2015.

[16] S Mohammad Mavadati, Mohammad H Mahoor, Kevin Bartlett, and Philip Trinh. Automatic detection of non-posed facial action units. In *2012 19th IEEE International Conference on Image Processing*, pages 1817–1820. IEEE, 2012.

[17] P Ekman, E Rosenberg, and J Hager. Facial action coding system affect interpretation dictionary (facsaid). *In.*, 1998.

[18] Maja Pantic and Leon JM Rothkrantz. Facial action recognition for facial expression analysis from static face images. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(3):1449–1461, 2004.

[19] Deepak Ghimire and Joonwhoan Lee. Geometric feature-based facial expression recognition in image sequences using multi-class adaboost and support vector machines. *Sensors*, 13(6):7714–7734, 2013.

[20] Patrick Lucey, Jeffrey F Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain Matthews. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *2010 ieee computer society conference on computer vision and pattern recognition-workshops*, pages 94–101. IEEE, 2010.

[21] Alexander AS Gunawan et al. Face expression detection on kinect using active appearance model and fuzzy logic. *Procedia Computer Science*, 59:268–274, 2015.

[22] Mariusz Szwoch and Pawel Pieniazek. Facial emotion recognition using depth data. In *2015 8th International Conference on Human System Interaction (HSI)*, pages 271–277. IEEE, 2015.

[23] SL Happy, Anjith George, and Aurobinda Routray. A real time facial expression classification system using local binary patterns. In *2012 4th International conference on intelligent human computer interaction (IHCI)*, pages 1–5. IEEE, 2012.

[24] Heechul Jung, Sihaeng Lee, Junho Yim, Sunjeong Park, and Junmo Kim. Joint fine-tuning in deep neural networks for facial expression recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 2983–2991, 2015.

[25] Behzad Hasani and Mohammad H Mahoor. Facial expression recognition using enhanced deep 3d convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 30–40, 2017.

[26] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

[27] Byoung Chul Ko. A brief review of facial emotion recognition based on visual information. *sensors*, 18(2):401, 2018.

[28] Deepak Kumar Jain, Zhang Zhang, and Kaiqi Huang. Multi angle optimal pattern-based deep learning for automatic facial expression recognition. *Pattern Recognition Letters*, 139:157–165, 2020.

[29] MAH Akhand, Shuvendu Roy, Nazmul Siddique, Md Abdus Samad Kamal, and Tetsuya Shimamura. Facial emotion recognition using transfer learning in the deep cnn. *Electronics*, 10(9):1036, 2021.

[30] Sanoar Hossain, Saiyed Umer, Vijayan Asari, and Ranjeet Kumar Rout. A unified framework of deep learning-based facial expression recognition system for diversified applications. *Applied Sciences*, 11(19):9174, 2021.

[31] Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9, 2010.

[32] James J Gibson. The perception of the visual world. 1950.

[33] David Fleet and Yair Weiss. Optical flow estimation. In *Handbook of mathematical models in computer vision*, pages 237–257. Springer, 2006.

[34] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003.

[35] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.

[36] Davis E King. Max-margin object detection. *arXiv preprint arXiv:1502.00046*, 2015.

[37] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE signal processing letters*, 23(10):1499–1503, 2016.

[38] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[39] Michael J Lyons, Miyuki Kamachi, and Jiro Gyoba. Coding facial expressions with gabor wavelets (ivc special issue). *arXiv preprint arXiv:2009.05938*, 2020.

[40] Michael J Lyons. " excavating ai" re-excavated: Debunking a fallacious account of the jaffe dataset. *arXiv preprint arXiv:2107.13998*, 2021.

[41] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[42] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[43] M Tan and Q EfficientNet Le. Rethinking model scaling for convolutional neural networks. arxiv 2019. *arXiv preprint arXiv:1905.11946*, 2020.

[44] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. 2015.

[45] Zhengyao Wen, Wenzhong Lin, Tao Wang, and Ge Xu. Distract your attention: Multi-head cross attention network for facial expression recognition. *arXiv preprint arXiv:2109.07270*, 2021.