POLITECNICO DI TORINO

College of Computer Engineering, Cinema and Mechatronics

Master's Degree Thesis

# Automatic Recognition And Classification Of Passengers' Emotions In Autonomous Driving Vehicles

**Supervisor:**
prof. Massimo VIOLANTE
**Co-Supervisor:**
dr. Jacopo SINI

**Candidate:**
Antonio Costantino MARCEDDU

Academic Year 2018-2019

# Summary

The advent of autonomous driving cars bring hopes of reducing automobile accidents, improving, in general, the road safety. Many problems remain to be solved, but technology has made great steps in these years.

A problem, that must not be underestimated, is how the people will react to these vehicles. In fact, the sense of security that is commonly felt is often false and guided by false certainties. Then, an important question, to which those working in the sector are trying to give an answer, is the following: *people will consider the self-driving cars safer than a vehicle driven by a human being, knowing that, for definition, a real self-driving car is completely independent and not subject to influences?* Researchers, therefore, continue to look for new ways to improve the social acceptance of these vehicles and the way they are seen by people.

With this thesis, we want to propose a new method that goes in this direction: *our idea is to monitor the mood of the people in the passenger compartment of a self-driving car, in order to change the driving style of the car itself.* For example, if people inside the car do not feel safe, thus feeling emotions of fear or sadness, the car will slow down its speed and will take the curves more slowly and with more precision, even coming to stop if necessary, when these feelings do not calm down within a certain period of time.

Due to the difficulty of access to autonomous driving cars, we started this project as a feasibility study, analyzing all aspects of it. We thought that mood recognition could be done using a neural network capable of recognizing the facial expressions of people inside the car. So, as first step, we have developed a new program, called *Facial Expressions Databases Classifier* (*FEDC*), to facilitate the preprocessing of the images necessary for the training of a neural network; we then used it to train two *state-of-the-art neural networks*, expressly created to detect facial expressions, with different databases, and which we later used to do some *road tests*, necessary to analyze the performance of the previously trained neural networks in real conditions. The road tests were carried out with another program that we developed specifically, called *Emotion Detector*, which we used to document the data coming from the neural networks.

From this study, we have obtained very interesting results, which have given us the determination to move on to the next phases of the project, and which will concern the implementation of the same into a single-board computer and a continuous improvement of FEDC, adding new features and supporting new facial expressions databases.

A partial work, derived from this thesis, was presented at the *Italian Workshop on Neural Networks* (*WIRN*) *2019*, held from June 11th to 14th in Vietri sul Mare (SA), Italy, and we are awaiting a response for publication in the book related to the conference, published by Springer.

# Acknowledgements

I would like to thank prof. Massimo Violante and dr. Jacopo Sini for their support during the realization of this thesis.

I am grateful to my love Sandra, who supported me along my university path, giving me the strength to overcome all the problems I encountered, and which continues to support me every day.

I am also grateful to my father Graziano, to my mother Silvia, to my sister Valentina and to my grandmothers Amalia and Maria, who have provided me, throughout my life, with moral and emotional backing.

Finally, I would like to thank my other family members, friends, and colleagues who have supported me along the way.

Page intentionally left blank.

# Contents

# List of Figures

# List of Tables

Page intentionally left blank.

Page intentionally left blank.

# Chapter 1

# Introduction

## 1.1 Background

**Cars** are means of transport that, in our contemporaneity, fully represent the ideal of **personal freedom**; unlike trains and trams, where it is normal for the vehicle to be always driven by a third party, cars can be owned and driven by everyone, and are commonly used to carry out normal daily activities, such as going to work, shopping, gym and so on. The cars entered our lives a long time ago, so, nowadays, for a young boy or a young girl, it is normal to think of wanting to buy one, especially after obtaining the driving license.

The cars began to have affordable prices at the beginning of the XX century, a time when Ford dominated the sector and was able to produce vehicles, like the Ford Model N and the Ford Model T, which cost about half or less than the average annual wage in the United States. A long way has been made since then, and safer cars, full of optionals and more eco-friendly, can be purchased by anyone [45].

In recent decades, there has been a rapid evolution, that made some forms of transport, generally of terrestrial type, semi or fully automated. The automation progress has gone in parallel for rail vehicles, like trains and trams, and for non-rail vehicles, primarily autos. Nowadays, the first types of vehicle, in all its forms of automation, also the most advanced ones, is rather common and they are diffused all around the globe; for the second type, however, only lower types of automation are common, because they present more complexities and ethical problems than the former. Fully automated driving cars are being tested, although, for now, it is necessary to have a human supervisor who can take control of the car in extraordinary emergency procedures in which the automatic system fails, in order to avoid the violation of national laws.

But **what is commonly meant by the term "autonomous driving car"**? For UCSUSA [55], "self-driving vehicles are cars or trucks in which human drivers are never required to take control to safely operate the vehicle". For Treccani [54], the automatic driving regards the "driving and control of robotic vehicles through environmental data acquisition systems and automatic driving regulation, without the need for intervention by a human driver"[1]. For Techopedia [53], "an autonomous car is a vehicle that can guide itself without human conduction".

As we can read from the different definitions, autonomous driving cars are described as fully automated, without human intervention. This is an optimum case: an external control of the car can still be indispensable, at least for now, in certain situations, like in the event of a traffic accident or when you need to park your car in large garages.

One of the most important questions that car manufacturers, car rental agencies and other actors are trying to answer is: **will people put their trust in autonomous driving cars?** It is a question that is not easy to answer, because the sense of security that is commonly felt is often false and guided by false certainties; new technologies and methods to improve the social acceptance for this type of vehicles are essential for the answer to be affirmative.

Therefore, for this thesis, we have developed a project that could be helpful in this direction, and that we will present in the next section.

## 1.2 Our idea

We thought that **a system in which the car is able to react according to the passengers' emotions, changing its driving style, could be useful for the purpose**. So, we asked ourselves: **what emotions could be useful as input for the vehicle?** These emotions are expressed in the same way by everyone, or they can change based on cultural, individual or other factors? In the early 1970s, psychologist Paul Ekman answered this question by identifying six basic emotional states, common to all cultures [3]:

- anger;
- disgust;
- fear;
- happiness;
- sadness;
- surprise.

[1]Translated by the author.

In later years, Paul Ekman himself and other researchers continued this study, finding other common emotions, of which we will not discuss [16] [1]. So, we thought that four of these emotions, namely fear, happiness, sadness and surprise, with the addition of the neutrality, could actually be used as input, for the car, in this way:

- if most of the people inside the car experience feelings of **fear** or **sadness**, the car will slow down its speed and take curves more accurately, with a lower lateral acceleration, even coming to stop if necessary, when these feelings do not calm down within a certain period of time;
- if most of the people inside the car experience **neutral** feelings, the car will use its standard parameters;
- if most of the people inside the car experience feelings of **happiness** or **surprise**, the car will go faster and will have a more sporting trim, always respecting the speed limits of the road in which it is located.

In case of need, if among the people in the passengers compartment there is no majority state of mind, it is possible to proceed with strategies different from those normally provided, such as, for example, the assignment of a higher priority to passengers in state of fear, the use of the value obtained previously and others.

We thought about how we could actually implement this technology: for this purpose, after drawing some sketches, we made a 3D rendering, which you can see in figure 1.1.

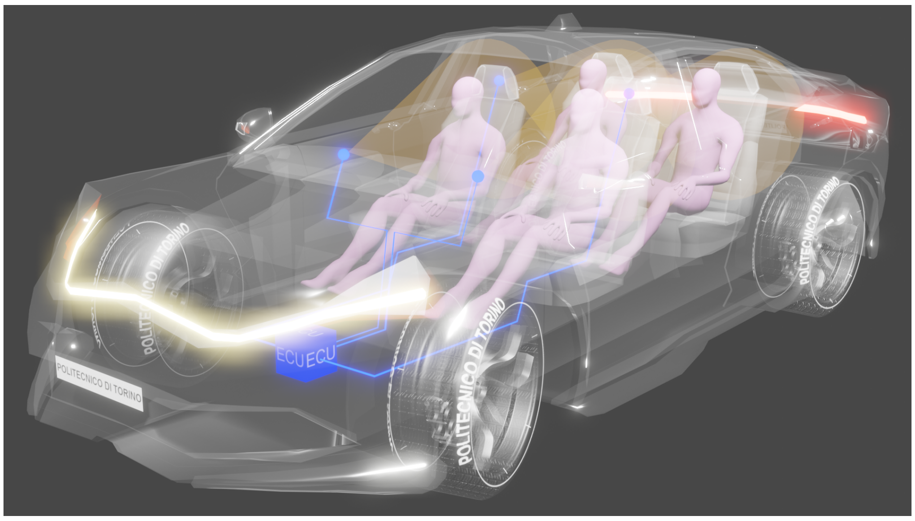


Figure 1.1: A possible implementation of this project in an autonomous driving car.

This system could work with a properly trained neural network to recognize facial expressions. Nowadays, they are the standard for carrying out this type of activities.

## 1.3 Thesis structure

We got the idea and we defined how the system could work; the next step was to define a **roadmap** to follow during the thesis. For now, it is difficult to access to an autonomous driving vehicle, so we decided, as a first step, to do a feasibility study, in order to know if this approach can actually work. To this end, we have developed and followed this roadmap:

- obtain information about the state of the art regarding the **neural networks capable of making facial expressions recognition**. We will discuss this in the third chapter, called *Training of neural networks*;
- gain access to some **facial expressions databases**, in order to have the material needed to train a neural network, and find an intelligent way to **preprocess** these databases in order to be a good input for it. We will discuss this in the second chapter, called *Preprocessing*;
- **train a neural network**, able of recognizing facial expressions with a good accuracy, taking into account the fact that it must work efficiently in different light conditions, so it is very important that it is able to generalize. We will discuss this in the third chapter, called *Training of neural networks*;
- **perform some road tests**, in order to test some of the previously trained neural networks. We will discuss this in the fourth chapter, called *Road tests*.

A partial work, derived from this thesis, was presented at the Italian Workshop on Neural Networks (WIRN) 2019, held from June 11th to 14th in Vietri sul Mare (SA), Italy (figure 1.2), and we are awaiting a response for publication in the book related to the conference, published by Springer.

State of art
Passengers are not used to driverless vehicles.
Different market model (ownership, rent, taxi-like service)
Humans show to others their emotions through facial expressions controlled by facial muscles.
Databases of facial expressions are available in literature

Figure 1.2: A photo taken at the Italian Workshop on Neural Networks (WIRN) 2019.

# Chapter 2

# Preprocessing

*For the first part of our work, we focused on the preprocessing phase. We have solved the related problem with a program: before talking about it, we want to discuss about how the idea of realizing this program came from.*

## 2.1 Previous works

In 2018, I participated in the “Smart Mood Reader” Hackathon, that took place in the CLIK (Contamination Lab and Innovation Kitchen) laboratory of the Politecnico di Torino. For this hackathon, I worked together with Mirko Crobu, Jessica Contreras and Daniel Gaiki for the development of a project called **OVU**, an acronym that came from the Italian words Olfatto (smell), Vista (sight), and Udito (hearing).

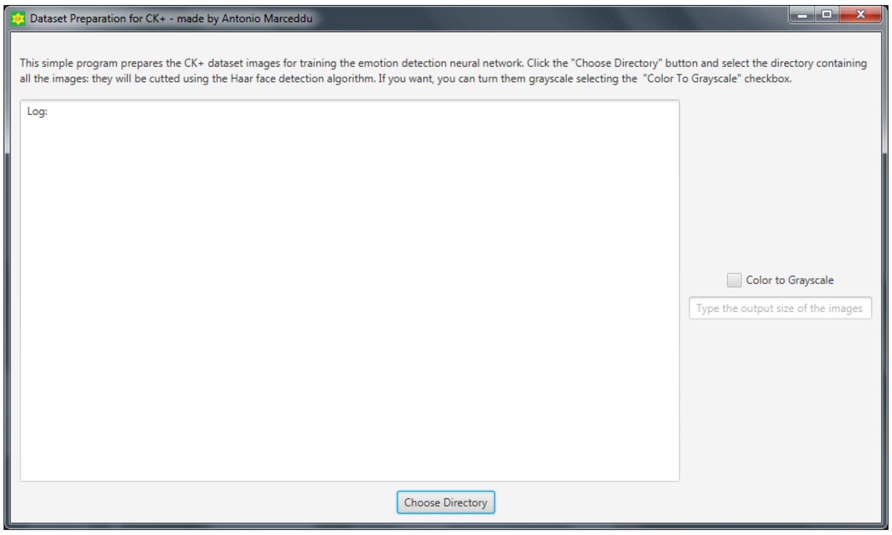
Figure 2.1: The OVU team on the proclamation day of the winner of the hackathon.

OVU is a system that combines sounds, images and colored lights in order to improve the mood of a person: for example, if a person is angry, OVU recognizes its mood and displays on a TV or a smartphone a relaxing three-dimensional musical video, where the user's interactions can change the color of the lights in the room where he is; the colors are defined according to their effects deriving from studies on chromotherapy, and they are combined with the right images and sounds in order to improve their effect. With this project we achieved the first position, winning the hackathon (figure 2.1). The reward was the possibility to expose our project at the Maker Faire of Rome and Turin, so we brought OVU there, where we got very useful ideas and impressions (figure 2.2).

Figure 2.2: The OVU stand at the Turin Maker Faire.

OVU consisted of two parts: the physical part, composed of a **Raspberry Pi model 3** in which is executed a **Python** software, capable of capturing images using a camera and requesting the mood detection, and by a pure software part, realized with **Unity**, that receives the captured mood and that displays, consequentially, a scene. A curiosity about OVU is that this name, in the Sardinian language, means "egg": therefore, we decided to make the shell of the physical part of OVU with an ovoid shape, in order to play with this homonym.

The mood detection in OVU was carried out with the **Microsoft Azure API**: however, we thought it could be better, to protect users' privacy, to do this on-site; this also has the advantage of not having to depend on external services anymore, and, potentially, of doing the operation faster and with greater reliability of response than the counterpart obtainable with a cloud service. So, for the exam of Computer Vision, which required the realization of a project, we proposed and get permission to realize a neural network able to recognize facial expressions.

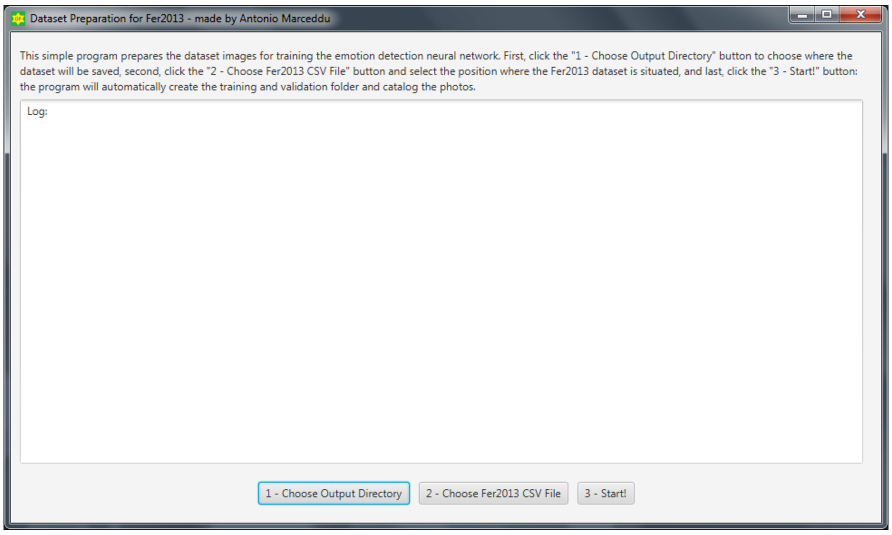

Dataset Preparation for CK+ - made by Antonio Marceddu

This simple program prepares the CK+ dataset images for training the emotion detection neural network. Click the "Choose Directory" button and select the directory containing all the images: they will be cutted using the Haar face detection algorithm. If you want, you can turn them grayscale selecting the "Color To Grayscale" checkbox.

Log:

Figure 2.3: Dataset Preparation for CK+.

In this project, we had to solve the same preprocessing phase problems that we had in this thesis, so we created two programs, called **Data Preparation for CK+** (figure 2.3) and **Data Preparation for FER2013** (figure 2.4): these programs, created resorting to Eclipse with the addition of **Java** and **OpenCV** [9], classify the images putting them in the right classification folder, and the CK+ [20] [24] version can also perform operations like grayscale conversion and image resizing, all with a simple and effective GUI.


Dataset Preparation for Fer2013 - made by Antonio Marceddu

This simple program prepares the dataset images for training the emotion detection neural network. First, click the "1 - Choose Output Directory" button to choose where the dataset will be saved, second, click the "2 - Choose Fer2013 CSV File" button and select the position where the Fer2013 dataset is situated, and last, click the "3 - Start!" button: the program will automatically create the training and validation folder and catalog the photos.

Log:

Figure 2.4: Dataset Preparation for FER2013.

So, when we got this thesis, we proposed and get the permission to create a program that included the functionality of these two programs, but that supported more databases of facial expressions and more functions.

## 2.2 Posed and spontaneous facial expressions

We want to make a computer able to classify facial expressions [4]. To obtain this result, therefore, a deterministic methodology, in which a facial expression can be clearly determined, is needed.

A commonly used system, capable of classifying them, is the **Facial Action Coding System** (**FACS**) [2]. The premise is that our face is full of muscles: every single type of movement that can be done by a facial muscle or a group of them, in the FACS system, is called **Action Unit** (**AU**). Thus, facial expressions can be adequately classified, using this system, as groups of weighted AUs. There may be more muscle configurations that can lead to defining a facial expression as part of a particular class. For example, a person can be considered happy if he has a beautiful smile arched upwards, but even if he has an open mouth and exposes his teeth; if frowning, however, this last expression could also be associated with a state of anger. Therefore, among the various facial expressions, there may be a strong intraclass variability that can, in some way, undermine the accuracy of the recognition implemented by an automated device.

The facial expressions databases make a clear distinction between **posed** and **spontaneous** facial expressions: while the latter are the natural response to particular stimuli, the former are, in general, a more pronounced version of them.

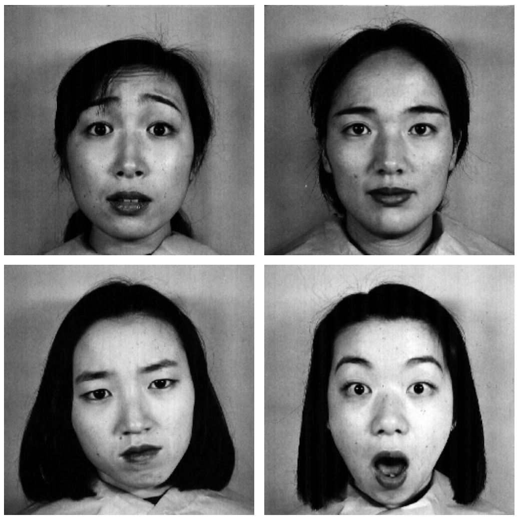


Figure 2.5: Example of photos taken from the JAFFE database.

## 2.3 Facial Expressions Databases Classifier

The program that we proposed and then created, always resorting to **Eclipse** with the addition of **Java** and **OpenCV** [9], was **Facial Expressions Databases Classifier** (**FEDC**) [48]; the main idea behind FEDC is the following: reduce to the minimum the time and the code necessary for the classification of facial expressions databases. Currently, FEDC is at version 4.0.3: starting from version 1.0.0, FEDC is able to automatically classify images of some of the most used databases, depicting human faces with posed and spontaneous expressions:

- **Extended Cohn-Kanade Database** (**CK+**) [20] [24];
- **Facial Expression Recognition 2013 Database** (**FER2013**) [18];
- **Japanese Female Facial Expression Database** (**JAFFE**) [25];
- **Multimedia Understanding Group Database** (**MUG**) [7];
- **Radboud Faces Database** (**RaFD**) [22];

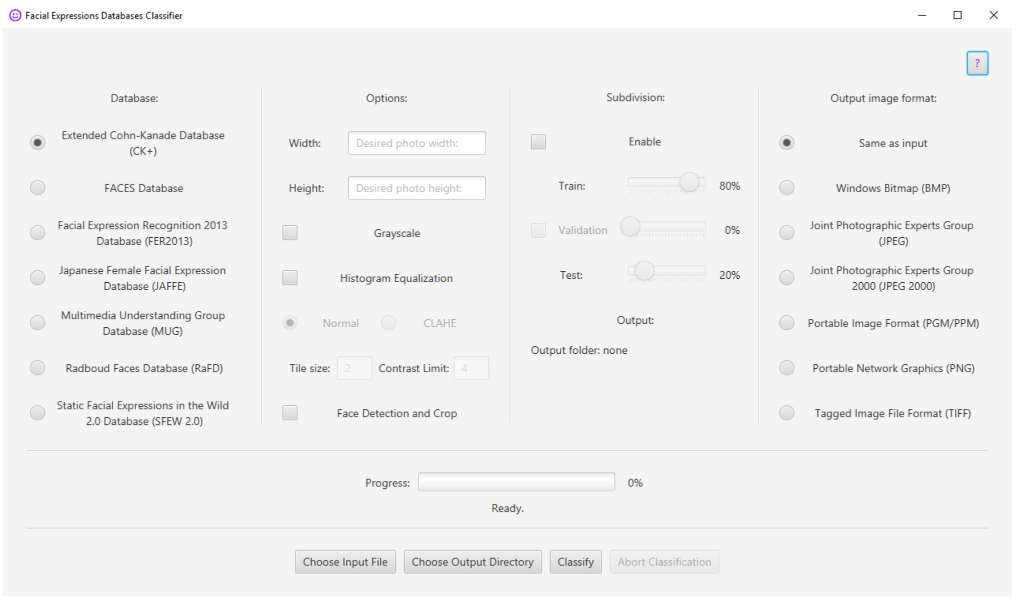

Facial Expressions Databases Classifier
?
Database:
Extended Cohn-Kanade Database (CK+)
FACES Database
Facial Expression Recognition 2013 Database (FER2013)
Japanese Female Facial Expression Database (JAFFE)
Multimedia Understanding Group Database (MUG)
Radboud Faces Database (RaFD)
Static Facial Expressions in the Wild 2.0 Database (SFEW 2.0)
Options:
Width: Desired photo width:
Height: Desired photo height:
Grayscale
Histogram Equalization
Normal
CLAHE
Tile size: 2
Contrast Limit: 4
Face Detection and Crop
Subdivision:
Enable
Train: 80%
Validation 0%
Test: 20%
Output:
Output folder: none
Output image format:
Same as input
Windows Bitmap (BMP)
Joint Photographic Experts Group (JPEG)
Joint Photographic Experts Group 2000 (JPEG 2000)
Portable Image Format (PGM/PPM)
Portable Network Graphics (PNG)
Tagged Image File Format (TIFF)
Progress: 0%
Ready.
Choose Input File
Choose Output Directory
Classify
Abort Classification

Figure 2.6: Facial Expressions Databases Classifier, version 4.0.3.

Starting from version 3.0.0, support to the **FACES Database** [15] was added, and since version 4.0.0, support to the **Static Facial Expressions in the Wild 2.0 Database** (**SFEW 2.0**) [12] was added, for a total of seven facial expressions databases supported. FEDC does nothing magical: it uses the pre-classification implemented by the databases' creators. In addition to this, FEDC, from version 1.0.0, is also able to perform several useful operations on images:

- conversion in grayscale color space;
- histogram equalization;
- face detection to crop the images to face only;
- scaling of horizontal and vertical resolutions.

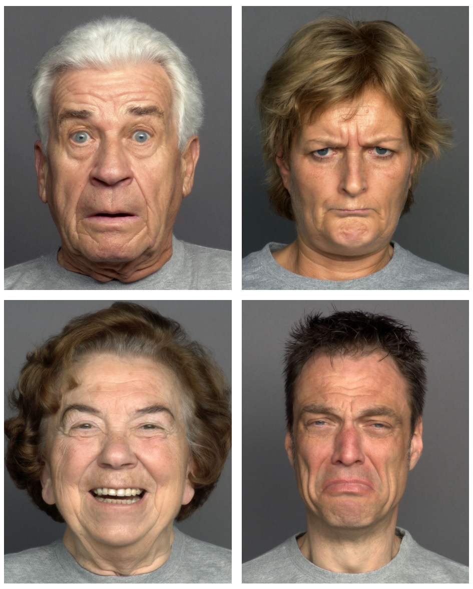


Figure 2.7: Example of photos taken from the FACES database.

Starting from version 2.0.0, the subdivision option between train, validation and test dataset was added; from version 3.0.0, the possibility of changing the output images format was added, and, from version 4.0.0, CLAHE (Contrast Limited Adaptive Histogram Equalization) option was added. Other useful additions, starting from version 4.0.0, regards the support to the **FER+ annotations** [8], of other two image formats, .PGM and .PPM, and of the option to add a padding in order to transform the rectangular images into square ones; this feature was initially released only for the SFEW 2.0 database, but, from version 4.0.2, it was extended to CK+, FACES and RaFD databases. This feature was a useful addition, because is common to build neural networks that work with square images.

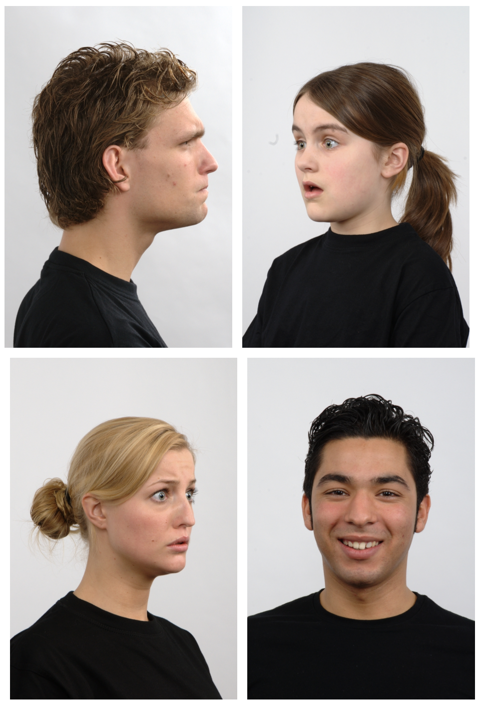


Figure 2.8: Example of photos taken from the RaFD database.

### 2.3.1 The FEDC interface

From the figure 2.6, it can be seen that FEDC [48] has a simple, clean and essential user interface, consisting of four macro-areas:

- on the left column, it is possible to choose the database to be classified, one between those listed previously;
- on the other columns, it is possible to select the operations that will be carried out on the photos: those available have already been mentioned above;
- on the lower part of the window, there are the buttons to select the input file, output folder, and for start and cancel the classification;
- finally, above the buttons, there is the progress bar and the state of the current operation.

The interface has changed considerably between the different versions, in order to accommodate all the new features: if you look at the figure 2.9, which represents FEDC version 1.0.0, you will notice that the new version is bigger and with more options.

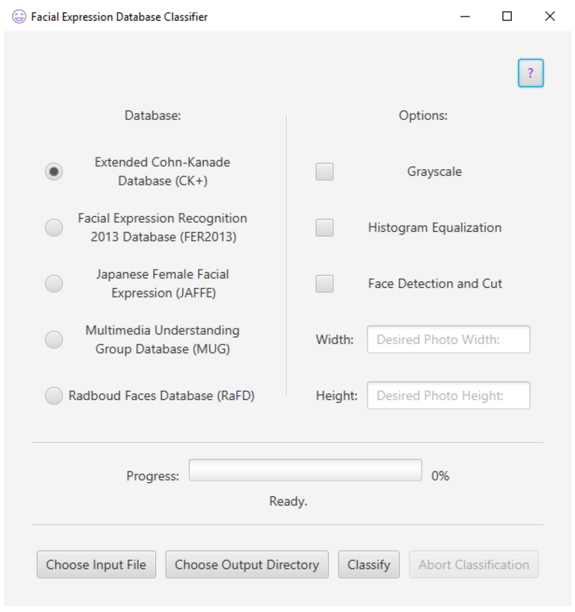

Facial Expression Database Classifier
?
Database:
Options:
Extended Cohn-Kanade Database (CK+)
Facial Expression Recognition 2013 Database (FER2013)
Japanese Female Facial Expression (JAFFE)
Multimedia Understanding Group Database (MUG)
Radboud Faces Database (RaFD)
Grayscale
Histogram Equalization
Face Detection and Cut
Width:
Desired Photo Width:
Height:
Desired Photo Height:
Progress:
0%
Ready.
Choose Input File
Choose Output Directory
Classify
Abort Classification

Figure 2.9: Facial Expressions Databases Classifier, version 1.0.0.

### 2.3.2 Database Ensemble

One of the features of FEDC [48] is to easily accomplish what we have called **Database Ensemble**, which is a union of multiple databases, containing the same classes or a subset of them. This merge operation can:

- increase the generalization capability of the neural network;
- partially or totally compensate the problems of having small databases.

For databases containing images, it is possible to uniform these data as much as possible, performing, for example, operations like grayscale conversion, histogram equalization and normalization: we will talk about this operation in the next chapter.

With FEDC, creating a Database Ensemble of facial expressions is simple: you must set in FEDC, as far as possible, the same parameters for each database that will be used, letting it prepare the images and then manually combining the obtained results. If you use the subdivision function of FEDC, you can obtain a train, test and, optionally, a validation dataset containing photos from all the databases, in order to have a reliable evaluation of its performance.

### 2.3.3 How FEDC works

Supported databases have features that make them different from one another; therefore, FEDC [48] will perform different operations for each of them. So, now we will make some examples regarding the operations carried out by FEDC for some of them.

### CK+ Case

The Cohn-Kanade Extended (CK+) database [20] [24] is composed of 593 sequence of 123 subjects, where, for each of them, the portrayed person starts from a neutral state and reaches a peak emotional state, which is in the last photo of each sequence. In this database are reported a total number of eight different facial expressions: anger, contempt, disgust, fear, happiness, neutrality, sadness and surprise. Of 593 peak emotional state photos, only 327 are FACS coded [2]: so, only these are relevant if we want posed human faces. By adding a neutral photo for every subject, which we said before being 123, we can reach a quantity of 450 photos: this is exactly the number of photos that FEDC [48] will return. Others have chosen a different number of photo, from the database, to train a neural network, for example the first and the last three photos of each sequence [17]: for FEDC, we have chosen to use only the FACS coded photos and the first photo of each sequence, but, if you want, you can modify slightly the FEDC code, in order to achieve the desired results. The images are contained in the "extended-cohn-kanade-images.zip" file, while the emotion coded files are contained in the "Emotion_labels.zip" file: these are the input files required by FEDC. If given, it will perform the following operations:

- if the user has not chosen to crop the images to face only, it asks him if he wants to make square the images;
- extracts the images and the labels contained in the two .zip files in a temporary folder;
- creates the classification folders and, in case, the subdivision folders;
- for every emotion coded file, checks if the file name is the one expected; if true, it reads it, obtaining the FACS coded emotion, and searches the corresponding image;
- once found, applies any optional operations selected by the user to the image (grayscale color space conversion, histogram equalization, etc.);
- saves the image, with the chosen format, in the corresponding classification folder.
- for every subject, adds the first photo of a sequence to the neutrality classification folder;
- optionally, if chosen by the user, subdivides the images between train, test and, always optionally, validation folders;

- deletes the temporary folders.

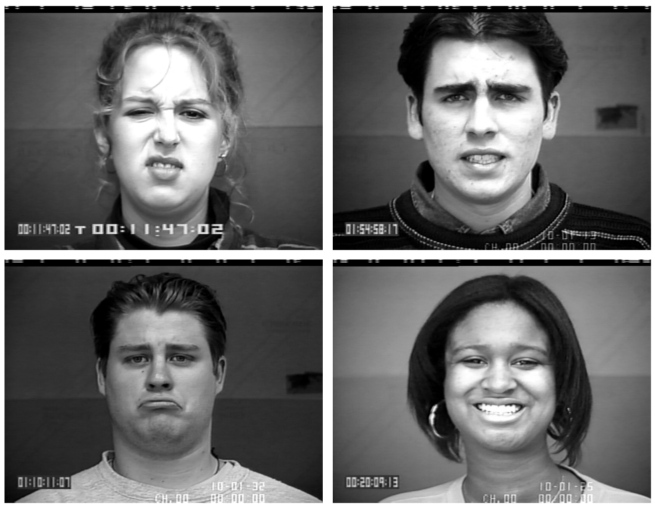


Figure 2.10: Example of photos taken from the CK+ database.

### FER2013 Case

The FER2013 database [18] is composed of 35887 photos, which depicts people with seven different spontaneous facial expressions: anger, disgust, fear, happiness, neutrality, sadness and surprise. It presents a default subdivision between train, validation and test of 80%-10%-10%, so, as first thing, FEDC [48] will ask if you want to use it or not: if you do not want to use it and you have not chosen to use the FEDC integrated subdivision, the photos will be inserted in a single folder, and in the right classification folder.

The images and the information about the facial expressions and the subdivision folders are contained in a unique file, called "fer2013.csv". This is the input file required by FEDC. If given, it will perform the following operations:

- if the user has not chosen to perform the automatic subdivision made by FEDC, asks him if he wants to use the default subdivision of the database;
- creates the classification folders and, in case, the subdivision folders;
- starting from every byte of them, it recreates the images; if the number of bytes of one of them exceeds 2304, there will be an error, because FEDC expects images of 48x48 pixels;
- applies any optional user-defined operation: it is relevant to say that the grayscale option has no effect, because the images already are;

- saves the images, with the chosen format, in the right classification folder, eventually using the default subdivision super folder.
- if chosen by the user and if the default subdivision was not selected, subdivides the images between train, test and, always optionally, validation folders;
- deletes the temporary folder.

Optionally, you can also provide to FEDC the FER+ annotations [8], that re-label FER2013 adding also the contempt state and removing some images, contained in it, that do not contain human faces. The FER+ annotations are applied by FEDC using the "Majority Voting" modality: for more information, please read the FER+ annotations paper.

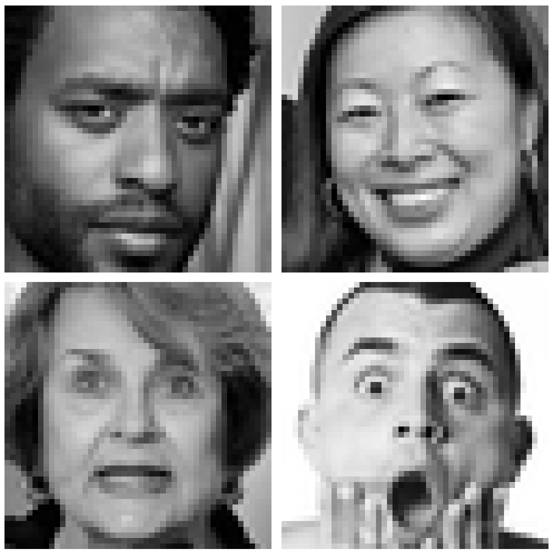


Figure 2.11: Example of photos taken from the FER2013 database.

### SFEW 2.0 Case

The Static Facial Expressions in the Wild (SFEW) 2.0 database [12] is composed of frames picturing actors with spontaneous expressions, extracted from Acted Facial Expressions in the Wild (AFEW) [13] [14], a database that contains sequence taken from different movies. SFEW contains aligned photos, that contains only the cropped faces of the actors, and non-aligned photos, that are the entire frames, both subdivided between train, validation and test folders. The number of aligned photos is 1694, divided as follows:

- 891 for train;

- 431 for validation;
- 372 for test.

Instead, the non-aligned photos are 1766, divided as follows:

- 958 for train;
- 436 for validation;
- 372 for test.

The test folders contains non-labeled photos, so they cannot be classified: however, the chosen transformations will also be applied to the test dataset, so as to give the user the freedom to use it, for example as a test for a neural network trained with this database. FEDC [48] works only with the aligned faces images: this because the rest of the frame is not useful for the evaluation of the facial expressions, so, at the moment, the non-aligned faces images are not supported. The aligned faces images are non-square; as we said above, with the introduction of SFEW 2.0, the option to add a padding to make these images square was added to FEDC, so the user can choose if he wants that the images become square or not. The aligned photos were cropped with automatic tools: therefore, there are images that do not contain human faces or that have a wrong crop. FEDC can eventually remove these photos.

For classifying the SFEW 2.0 database, FEDC requires these files: "Train-_Aligned_Faces.zip", "Val_Aligned_Faces_new.zip" and "Test_Aligned_Faces-.zip". If you input them to FEDC, it will perform the following operations:

- asks the user if he wants the images with an incorrect crop to be discarded;
- if the user has not chosen to perform the automatic subdivision made by FEDC, asks him if he wants to use the default subdivision of the database;
- asks the user if he wants to make square the images;
- extracts the images contained in the three .zip files in a temporary folder;
- if the user has not chosen the default subdivision, it creates the classification folders and, in case, the subdivision folders;
- applies the operations chosen by the user;
- saves the image, with the chosen format, in the corresponding classification folder;
- optionally, if chosen by the user, subdivides the images between train, test and, always optionally, validation folders;
- deletes the temporary folders.

Figure 2.12: Example of photos taken from the SFEW 2.0 database.

### 2.3.4 Information about supported databases

We have provided some information regarding the CK+ [20] [24], FER2013 [18] and SFEW 2.0 [12] databases, and we have seen how FEDC [48] works with them; also the other supported databases have particularities, which we would like to discuss here:

- the JAFFE database [25], like the FER2013 database, only contain grayscale images;
- the RaFD database [22] also contains profile photos. If you enable the option to perform face detection and crop of the images, FEDC will ask if you want to use a separate, less accurate face detector for this type of photo; independently from the answer, FEDC will exclude photos in which it has not found any human face;
- regarding the MUG database [7], FEDC only works with the manual annotated images, contained in the "manual.tar" file;

Finally, it is important to add that FEDC requires the ownership of the previously cited databases to work; the access to these is usually allowed only for research purposes: for more information, consult the related sites.

### 2.3.5 Final notes about FEDC

We think that FEDC [48] is a very useful tool for people who work with facial expressions: it was designed with the word "flexibility" in mind, so its code was released under the MIT license on GitHub. Everyone can download it and use it freely, modifying the code at will. The code is all commented in english, so this simplifies understanding. The addition of a classifier for a new database is simple: there is a class, called Classifier.java, which implements all the optional operations to be applied on the images present in FEDC, and which can be inherited from the new classifiers and used, reducing the number of lines to write; also, unzipping and untarring classes are present. If desired, the code can be freely modified so that it can also be used in totally different contexts than that of facial expressions.

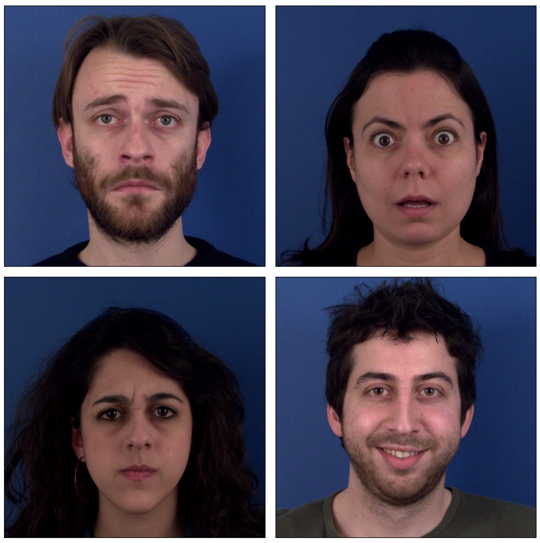


Figure 2.13: Example of photos taken from the MUG database.

# Chapter 3

# Training of neural networks

*After creating Facial Expressions Databases Classifier, we wanted to use it for training neural networks capable of recognizing facial expressions. Therefore, we looked for the state of the art regarding the neural networks created for this purpose, in order to make comparisons on the degree of accuracy that we have been able to obtain with respect to that obtained by others. It is important to note that we do not want to build a new high-performance neural network specifically for facial expressions, but we are rather interested in make experiments in such a way we know if we can use this technology for making predictions in cars interiors.*

## 3.1 Brief history of neural networks

The history of the neural networks started a long time ago, when studies on the behavior of the human brain began. But it is from the end of the XIX century that the most important phases took place: now, we will mention some of the most important events [51].

In 1943, Warren McCulloch and Walter Pitts wrote an article concerning the possible functioning of a neuron. This was an input for other researches, and, in 1958, Frank Rosenblatt defined the **Perceptron** (figure 3.1), a simplified mathematical model of a neuron. A year later, Bernard Widrow and Marcian Hoff developed the single-layer neural network **ADALINE** (acronym of **ADAptive LINear Elements**) and its multi-layer version **MADALINE**, which is still commercially used today.

After a period in which studies froze for various reasons, such as unfulfilled claims and fears, they resumed in the early 1980s. in 1997, Sepp Hochreiter and Jürgen Schmidhuber defined the **LSTM** (**Long-Short Term Memory**) recurrent neural network (RNN).

in 1998, Yann LeCun, Joshua Bengio and Patrick Haffner created the **LeNet-5** [23] convolutional neural network (CNN): originally, it was built to automatically read the digits on bank cheques in the United States, but is still used nowadays in industry.

After a long time, in which neural networks sporadically appeared in some contests and papers [10] [11], in 2012, exploiting the growing computational power of the new GPUs, a new neural network, called **AlexNet** [21], came out; it was presented at the **ImageNet Large Scale Visual Recognition Challenge 2012** (acronym **ILSVRC2012**), imposing on opponents with an error rate lower than almost 11% compared to them: an astonishing performance, which has catapulted the neural networks into the current trend topic in the field of data science.

The AlexNet document has been cited for more than forty thousand times, and continues to be a source of inspiration for the improvement of neural networks, which have greatly improved in these seven years.

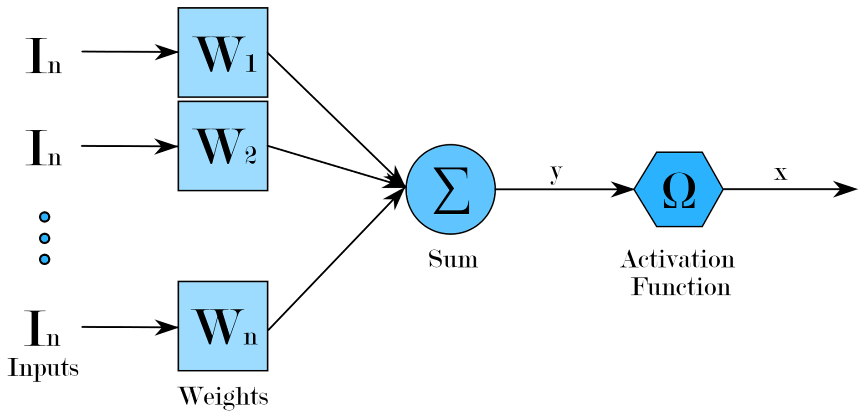


Figure 3.1: An image of the Perceptron.

## 3.2 Useful notions about neural networks

Before starting, it might be useful to mention some brief notions about neural networks, because we will use them abundantly later.

### 3.2.1 Definition

In the last section, we briefly talked about the history of neural networks. An **Artificial Neural Network** (**ANN**) [46] [56] is a mathematical system, partly inspired to the human brain. For the sake of brevity, most of the time the word "artificial" is not reported, but it would be important to mention it, because there is a huge difference between a normal neural network and an artificial one. An ANN is composed of artificial:

- **synapses**, which have a weighting and interconnection function;
- **neurons**, which makes the linear combination between the inputs coming from the synapses in this way:

$$\mathrm{Y} = \sum(\mathrm{weight} \cdot \mathrm{input}) + \mathrm{bias}$$

This value is then passed to an **activation function**, which is needed to decide if the neuron is activated or not. Some of most common activation functions are the following:

- **ELU**;
- **LeakyReLU**;
- **Linear**;
- **PReLU**;
- **ReLU**;
- **Sigmoid**;
- **Softmax**;
- **Tanh**.

For the neural networks that we will experiment in this thesis, we will use the ReLU, LeakyReLU and Softmax activation functions.

### 3.2.2 Overfitting and Underfitting

For a machine learning system, the primary objective is to create a model that is able to generalize well [41]: this means that the model, optimistically, must work in the same way both with data already seen and with new, unseen data. A model is called:

- **overfitted** when it learned too many features from the input data, thereby limiting its ability to generalize towards new data;
- **underfitted** when it has not been able to learn sufficient characteristics from the input data, and that, consequently, is not able to generalize towards new data.

A model that is capable to generalize in a good way will be neither overfitted nor underfitted.

### 3.2.3 Bias and Variance

A model, when trained, is able to make good predictions about the nature of a sample if:

- it belongs to one of the classes that the model can classify;
- it is sufficiently similar to the samples used for its training;

If the model had been trained with very different classes, the accuracy of the prediction is expected to be higher than if it is trained with very similar classes. So, it's important to understand two concepts regarding prediction errors [50]:

- the **bias** is the difference between the correct value that we are trying to predict and the average prediction. A high bias involves oversimplification and little attention to the important features of the training data;
- the **variance** indicates the sensitivity of the model to small perturbations of the train dataset. Contrarily to what we said for the case of high bias, when there is a high variance the model tends to learn too much from the training data, losing its ability to generalize well on the data it has never seen before.

Then, there will be overfitting when there is a high variance, while there will be underfitting when there is a high bias. Optimistically, one would like to have both a low bias and a low variance (figure 3.2).

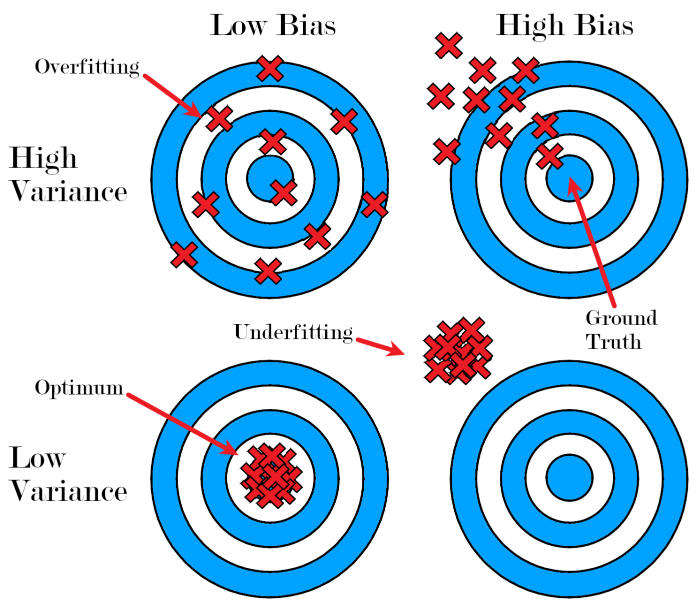


Figure 3.2: The Bias-Variance tradeoff.

### 3.2.4 Ground truth

The **ground truth** is the class that is assigned to a particular object by experts, and that assumes that this object actually belongs to that class. While for many problems understanding the class to which an object belongs can be simpler, as in the case of a binary problem of distinction between dogs and cats, for many other cases this may not be so simple, especially if the classes have minimal differences between them and have a very high acceptance variance, as happens, for example, exactly with facial expressions. The FER+ annotations paper [8] gives a good example in this regard; in fact, the team, after analyzing the entire FER2013 database [18], has efficiently documented the fact that:

- the ground truth sometimes is not exactly the truth;
- as the number of experts involved in making the classification increases, so too does the accuracy thereof.

### 3.2.5 Metrics

During the training, for evaluating the performance of a model are commonly used two metrics, that are the following:

- the **accuracy** [30] indicates the fraction of predictions guessed by the model, and it has the following formal definition:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

- the **loss** [31] indicates how bad the prediction of the model was on a single sample. The loss reaches zero if the prediction is perfect, otherwise it is greater than this value. There are different types of metrics; we will use the **categorical cross-entropy** [33], defined as:

$$L(y, \hat{y}) = -\sum_{j=0}^{M} \sum_{i=0}^{N} (y_{ij} \cdot \log(\hat{y}_{ij}))$$

### 3.2.6 Hyperparameters

The **hyperparameters** are variables that must be set before the training and that determines how the network will be trained. Now we will briefly discuss some of the most important hyperparameters [32]:

- the **learning rate** defines the speed of network updates; low values slow down the learning process but it becomes smoother, while high values accelerate the learning process, but increases the risk, for the network, of not having a convergence and to overfit;

- the **sample** is an element of a dataset. For example, it can be an image or an audio file;
- the **batch** is a set of N samples, processed independently and in parallel. During the training, a batch makes a single update of the network;
- an **epoch** is a limit that can be set by the user and is generally defined as "a passage on the entire dataset", used to divide the training in more phases in order to make, for example, periodic evaluation and logging. With Keras and similar tools, callback functions [37] can be called at the end of every epoch; we will make use of some of them, like, for example, the learning rate changer (ReduceLROnPlateau in Keras) and the early training stopper (EarlyStopping in Keras): these two callbacks are called when the given metrics have stopped improving;

### 3.2.7 Partitioning of the database

For the training a neural network, it is good practice to divide the database into three parts [36] [38]:

- the **train** dataset, that is used to train the network;
- the **validation** dataset, that is used to evaluate the network;
- the **test** dataset, that is used to test the network, in order to test the capability of the network to generalize on new, unseen data.

Percentages commonly used for the subdivision between train, validation and test datasets are 60%-20%-20% or 80%-10%-10% (figure 3.3).

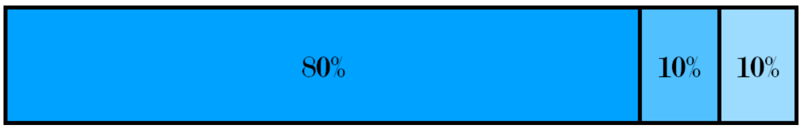


Figure 3.3: One of the possible subdivisions between train, validation, and test datasets: this configuration was the one we used for all the training we did.

Sometimes only train and test dataset, which also serves as validation dataset, are used, but this results in the following problem: in each epoch, the model makes predictions using the validation dataset to optimize his hyperparameters, but he does not learn from it: for this reason, if it is also used as test dataset, the error rate during the test will be lower than what really is. So, the golden standard requires the use of three distinct datasets.

A good technique to reduce the overfitting resulting from the training can be achieved partitioning the dataset into multiple subsets, in order to use one part for training and the remaining part for validation/testing, performing multiple rounds with different subsets of the same data so as to reduce the variance: this technique is called **cross-validation** [47]. There are several types of cross-validation, which we will discuss now:

- the **Leave-One-Out Cross-Validation** (**LOOCV**), which is performed by subdividing the dataset, that we suppose of size n, into two parts; one will contain one single observation, that will be used as validation/test dataset, and the other one will contain the rest of the dataset (n-1), which will be used for training. The operation will be repeated for every sample;

- the **k-Fold cross-validation**, which is performed, as the name suggests, splitting the dataset into k folds of approximately the same size, and using k-1 fold for training and the remaining for validation/testing (figure 3.4). The process will be repeated k times, modifying the groupings. The LOOCV is a variant of the k-Fold in which k is equal to n;

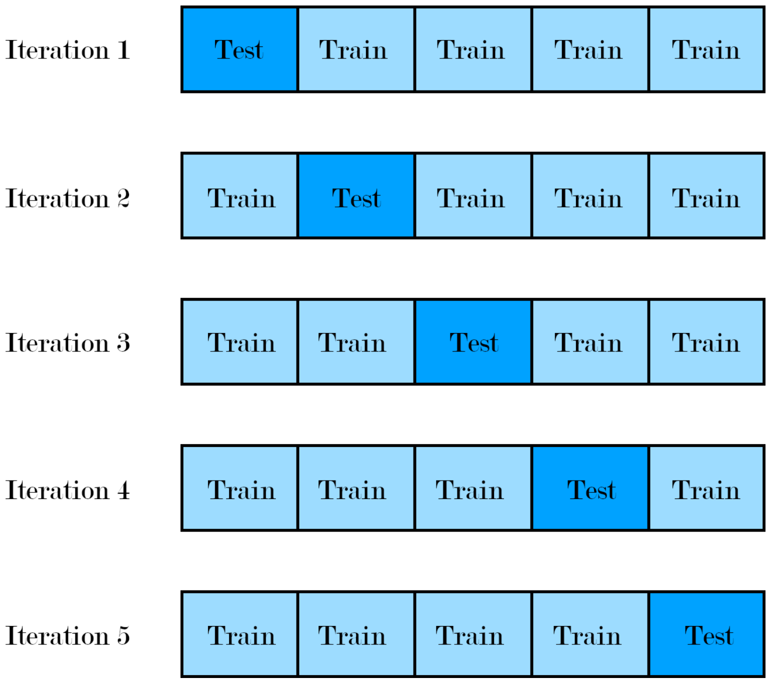


Figure 3.4: An example of 5-Fold cross-validation.

- the **Stratified cross-validation**, which is done by rearranging the dataset so that each fold can be seen as a good representation of the whole dataset. This is used for binary classification problems;

- the **Time-Series cross-validation**, that is used when the dataset is composed of observations made at different time points.

In general, cross-validation reduces bias and variance, at the cost of multiple training, which can be very expensive. For this thesis, we have chosen to use the k-Fold cross-validation for every test we have performed.

## 3.3 Other notions

In this section, we will briefly discuss about some operations and visualization methods that we will use later.

### 3.3.1 Normalization

The **normalization** [34] is a family of operations, commonly used in machine learning and other fields, that is used to bring the numeric features of a distribution on a similar scale. Commonly used normalization techniques are:

- the **feature clipping**, which is an operation, usually complementary to other types of normalization, used to set the interval of a distribution, cutting the values above it.
- the **log scaling**, which is an operation that compresses a wide range into a narrow one; this helps to improve the performance of a linear model. It is defined as follows:

  $$x' = \log(x)$$

- the **scaling to a range** normalization, which, as the name implies, changes the range of the distribution into the desired one. Common scaling range are $[-1, +1]$ and [0,1]. This is useful if the data are approximately uniform in a known range and there are some outliers or none of them. For scaling into a range, the following operation must be applied:

  $$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- the **z-score** normalization, also called **z-score standardization**, which is used to make sure that the distribution has a mean equal to zero and a standard deviation equal to one. It is useful when there is a limited number of outliers and their cropping is not necessary. It is defined as follows:

  $$x' = \frac{x - \mu}{\sigma}$$

We will make some tests using the scaling to a range normalization between 0 and 1, and the z-score standardization.

### 3.3.2 Data augmentation

The **data augmentation** [43] is a technique, commonly used in machine learning when you have a relatively small database, and that consists in the generation of batches of samples, modified with respect to the original ones contained in the database. For example, if the database contains images, the augmented images are obtained with mixes of transformations like translations, rotations, shearings and many others. This improves the ability of the model to generalize towards new data, and its performance.

### 3.3.3 Confusion matrix

The **confusion matrix** [6] is a two-dimensional matrix, having the class predicted by the classifier and the true class of the sample as dimensions. It is commonly used to display the performance of a classifier respect to a test dataset. If the classifier performs well, the majority of the values will be found in the main diagonal of the confusion matrix.

## 3.4 Neural networks for facial expressions

For our task, we searched for some of the best neural networks created specifically for facial expressions recognition; our choice fell on those presented in the following papers:

- **Physiological Inspired Deep Neural Networks for Emotion Recognition** [17];
- **Recognizing Facial Expressions Using a Shallow Convolutional Neural Network** [27].

Both networks are quite recent, released in 2018 and 2019 respectively, and while the first is sufficiently complex, the second is much simpler. They have approximately the same number of parameters, around 9 million, but while the first network has a default images size of 120x120 pixels, the second has as default size 48x48 pixels; the first one, setting a images size of 48x48 pixels, reduces the number of hyperparameters to just over 2 million, a much lower value than the previous one: this is due to the many max pooling layers present in the network, which performs images downsampling thus also reducing the number of parameters. This is an important fact, because if we want to use these networks on the cars, using fewer parameters means less demand for computational power, and less space required for store the neural network. So, we will examine their performances, using the following databases as input:

- CK+ (only for the first network) [20] [24];
- FER2013 [18] and FER2013 with FER+ annotations [8];
- a Database Ensemble composed of all the facial expressions databases supported by FEDC;
- a Database Ensemble composed of all the posed facial expressions databases supported by FEDC.

## 3.5 Used tools

In October 2018, Kaggle conducted a worldwide survey, aimed at data science professionals and involving 23,859 people. One of the most interesting results of this survey is the one concerning the most used programming languages, on a regular basis, for data science (figure 3.5): **Python**, with its 83%, is the most used, followed by SQL with 44% and R with 36%. The great advantage of Python, compared to other languages, consists in its richness of data science libraries and in the ease of performing even very complex operations.

That is why, for our task, we decided to use Python 3.7, alongside with the **Anaconda distribution**, which we used as interpreter, and **JetBrains Pycharm**, which we used as primary IDE. We have written the code necessary to train the neural networks with **Keras** [44], a high-level neural networks API, written in Python and able to run on top of three different popular frameworks for machine learning:

- **TensorFlow**;
- **CNTK** (**The Microsoft Cognitive Toolkit**);
- **Theano**;

Of these three, we decided to use TensorFlow [40] as backend. We also used the following libraries:

- **Matplotlib** [19]: a powerful, 2D plotting library for Python;
- **NumPy** [5] [29]: a BSD licensed Python package used for scientific computing;
- **OpenCV** (**Open Source Computer Vision Library**) [9]: an open-source, computer vision and machine learning software library for C++, Java, MATLAB and Python;
- **Pandas** [26]: an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools;
- **Scikit-learn** [28]: an open-source, BSD licensed Python library for data mining and data analysis.

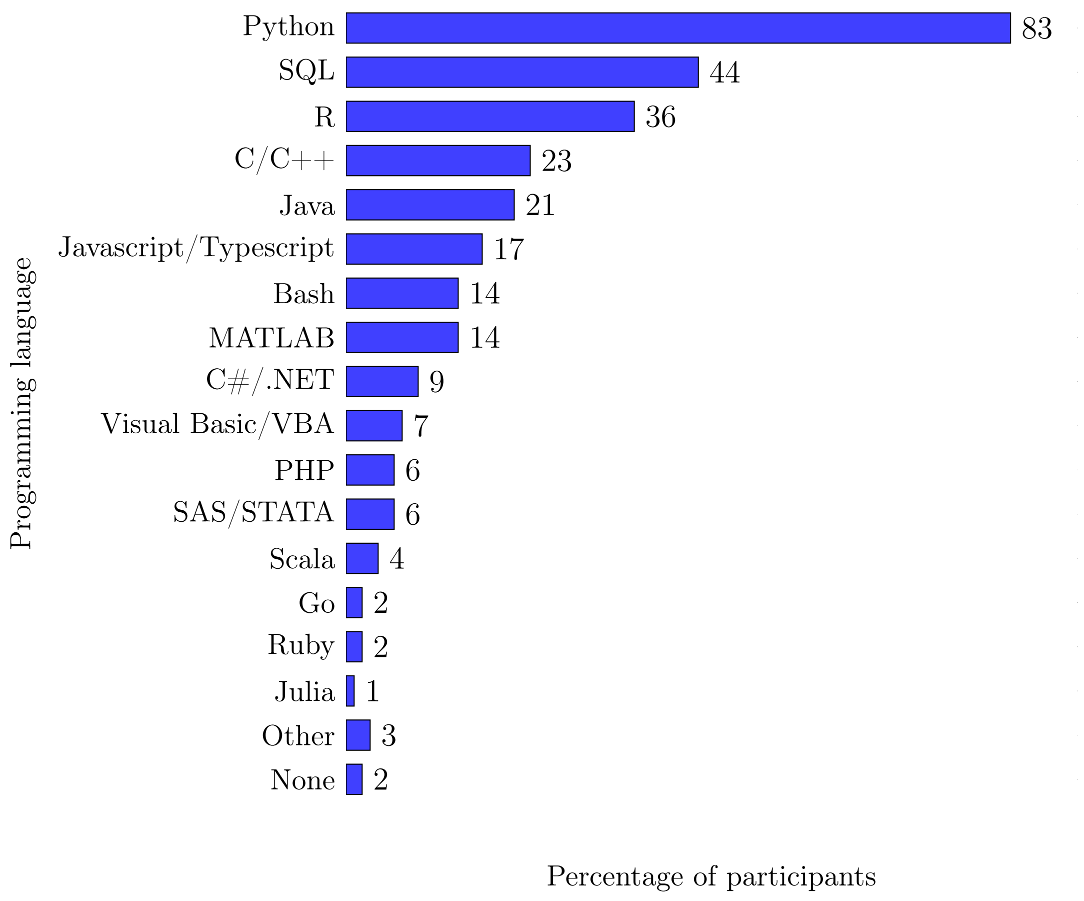


Figure 3.5: One of the most interesting results coming from the 2018 Kaggle Machine Learning & Data Science Survey.

## 3.6 Hardware configuration

For the training, we have used a laptop and a desktop computers with the following configurations:

- Laptop:

  CPU: Intel® Core™ i7 8750H;

  GPU: NVIDIA® GeForce® RTX 2080 8GB GDDR6 with Max-Q Design;

  RAM: 32 GB DDR4, 2666 MHz;

  256GB PCIe M.2 SSD + 1TB (+8GB SSHD) Hybrid Drive.

- Desktop:

  CPU: Intel® Core™ i7 4770K;

  GPU: NVIDIA® GeForce® GTX 1070 8GB GDDR5;

  RAM: 32 GB DDR3, 1866 MHz;

  1TB PCIe M.2 SSD + 512 GB SATA SSD + 1TB HDD.

In both cases, we used **TensorFlow-GPU** [40], which, for NVIDIA® graphic cards, recurs to the **NVIDIA® CUDA® Deep Neural Network** library (**cuDNN**) [35], which provides GPU-accelerated primitives for deep neural networks.

As much as possible, we have set some seeds to make the tests reproducible, but the use of a multicore CPU for training a neural network, or, better for performances, of a GPU, can introduce some sort of randomness in different part of the network [32] [42].

## 3.7 Trainings

### 3.7.1 Neural network from the "Physiological Inspired Deep Neural Networks for Emotion Recognition" paper

We have chosen to make our first tests with the network described in the **Physiological Inspired Deep Neural Networks for Emotion Recognition** paper [17], so we reimplemented it in Keras [44]. Some of the most interesting tests have been reported here.

For the e-block layer, we implemented the "crop and resize" operation, for the input coming from $E(x)$, using a single convolution layer: its kernel size has to be changed in base of the input images:

- for an images size of 120x120 pixels, that is the input default size for the network, a kernel size of 106x106 is needed;
- for an images size of 48x48 pixels, that we will use later, a kernel size of 43x43 is needed.

In both cases, we have set a number of output filters of 64, in order to make the inputs for the multiplication operation of equal size. For each test, we used the EarlyStopping callback, set up to block training if, after 20 epochs, there were no improvements to the validation loss, and the ReduceLROnPlateau callback, set to multiply the learning rate by 0.99 in every epoch.

| Layer # | Network Module | Layer (type) | Output shape | Connected to |
|---|---|---|---|---|
| 1 | $E(x)$ | input_1(InputLayer) | (3, 120, 120) | - |
| 2 | | conv2d_1(Conv2D) | (16, 120, 120) | input_1 |
| 3 | | conv2d_2(Conv2D) | (16, 120, 120) | conv2d_1 |
| 4 | | max_pool2d_1(MaxPooling2D) | (16, 60, 60) | conv2d_2 |
| 5 | | conv2d_3(Conv2D) | (32, 60, 60) | max_pool2d_1 |
| 6 | | conv2d_4(Conv2D) | (32, 60, 60) | conv2d_3 |
| 7 | | max_pool2d_2(MaxPooling2D) | (32, 30, 30) | conv2d_4 |
| 8 | | conv2d_5(Conv2D) | (64, 30, 30) | max_pool2d_2 |
| 9 | | conv2d_6(Conv2D) | (64, 30, 30) | conv2d_5 |
| 10 | | max_pool2d_3(MaxPooling2D) | (64, 15, 15) | conv2d_6 |
| 11 | | conv2d_7(Conv2D) | (128, 15, 15) | max_pool2d_3 |
| 12 | | conv2d_8(Conv2D) | (128, 15, 15) | conv2d_7 |
| 13 | | conv2d_tr_2(Conv2DTranspose) | (64, 30, 30) | conv2d_8 |
| 14 | | concatenate_1(Concatenate) | (128, 30, 30) | [conv2d_tr_1; conv2d_6] |
| 15 | | conv2d_9(Conv2D) | (64, 30, 30) | concatenate_1 |
| 16 | | conv2d_10(Conv2D) | (64, 30, 30) | conv2d_9 |
| 17 | | conv2d_tr_2(Conv2DTranspose) | (32, 60, 60) | conv2d_10 |
| 18 | | concatenate_2(Concatenate) | (64, 60, 60) | [conv2d_tr_2; conv2d_4] |
| 19 | | conv2d_11(Conv2D) | (32, 60, 60) | concatenate_2 |
| 20 | | conv2d_12(Conv2D) | (32, 60, 60) | conv2d_11 |
| 21 | | conv2d_tr_3(Conv2DTranspose) | (16, 120, 120) | conv2d_12 |
| 22 | | concatenate_3(Concatenate) | (32, 120, 120) | [conv2d_tr_3; conv2d_2] |
| 23 | | conv2d_13(Conv2D) | (16, 120, 120) | concatenate_3 |
| 24 | | conv2d_14(Conv2D) | (1, 120, 120) | conv2d_13 |
| 25 | $F(x, \hat{x})$ | conv2d_15(Conv2D) | (16, 120, 120) | input_1 |
| 26 | | conv2d_16 (Conv2D) | (16, 120, 120) | conv2d_15 |
| 27 | | max_pool2d_4(MaxPooling2D) | (16, 60, 60) | conv2d_16 |
| 28 | | conv2d_17(Conv2D) | (32, 60, 60) | max_pool2d_4 |
| 29 | | conv2d_18(Conv2D) | (32, 60, 60) | conv2d_17 |
| 30 | | max_pool2d_5(MaxPooling2D) | (64, 30, 30) | conv2d_18 |
| 31 | | conv2d_19(Conv2D) | (64, 30, 30) | max_pool2d_5 |
| 32 | | conv2d_20(Conv2D) | (64, 30, 30) | conv2d_19 |
| 33 | | max_pool2d_6(MaxPooling2D) | (64, 15, 15) | conv2d_20 |
| 34 | | eblock_1(e-block) | (64, 15, 15) | [max_pool2d_6; conv2d_14] |
| 35 | $G(h)$ | dense_1(Dense) | (512) | eblock_1 |
| 36 | | dropout_1(Dropout) | (512) | dense_1 |
| 37 | | dense_2(Dense) | (512) | dropout_1 |
| 38 | | dropout_2(Dropout) | (512) | dense_2 |
| 39 | | dense_3(Dense) | (8) | dropout_2 |

Table 3.1: The neural network described in the "Physiological Inspired Deep Neural Networks for Emotion Recognition" paper.

For every database, the initial configuration for our tests, with little modifications about the hyperparameters respect the ones reported in the paper, was the following:

- batch size: 50;
- maximum number of epochs: 1000;
- learning rate: 0.001;

- preliminary subdivision of 90%-10% between train and test dataset, made with FEDC [48];
- a 9-fold, which further divides the train dataset into variable train and validation datasets.

Through the preliminary subdivision and the 9-fold, the following subdivision of the database was obtained:

- 80% for the train dataset;
- 10% for the validation and the test datasets;

With FEDC, we converted to grayscale the images of the database, and choose to crop the images on the face only so as to remove the unwanted part of the images and to improve the results of the training.

### CK+ database

These first tests are useful to verify the goodness of the implemented network. The data distribution for the training with the CK+ database [20] [24] is reported in the figure 3.6. It should be noted that the total number of available images is very low and not well distributed among the different expressions: for example, it can be seen in fact that, for the contempt case, we have only 18 photos, against the 123 photos available for the neutrality case.

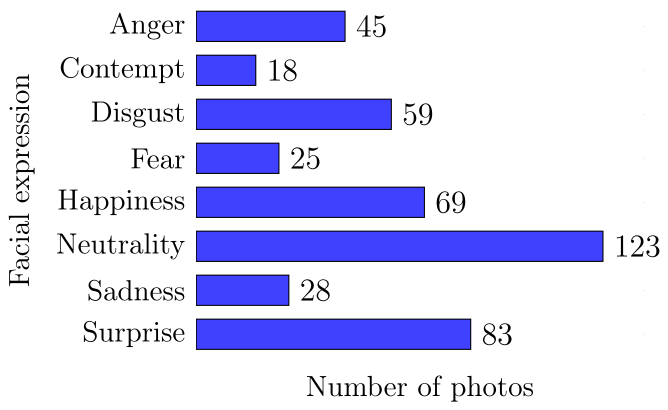


Figure 3.6: Subdivision of the images between classes in the CK+ database.

The subdivision between datasets, obtained with the settings listed above, is as follows:

- 364 photos for train;
- 46 photos for validation;
- 40 photos for test.

Using these settings, we trained 9 networks, obtaining the results in table 3.2: the best of them, achieved using the seventh fold as validation dataset, was capable to achieve a test accuracy of 82.5% after 137 epochs of training. It can also be noted that some networks have failed to converge at all.

| **Fold #** | **Epochs of Training** | **Test Accuracy** | **Max Validation Accuracy** | **Min Validation Loss** | **Final Validation Accuracy** | **Final Validation Loss** |
|---|---|---|---|---|---|---|
| 0 | 50 | 0.67500 | 0.78261 | 1.03806 | 0.69565 | 2.21111 |
| 1 | 65 | 0.80000 | 0.84783 | 1.09511 | 0.71739 | 1.74286 |
| 2 | 582 | 0.30000 | 0.21739 | 12.61581 | 0.21739 | 12.61581 |
| 3 | 32 | 0.37500 | 0.54348 | 1.82376 | 0.47826 | 3.19177 |
| 4 | 35 | 0.50000 | 0.60870 | 1.26470 | 0.58696 | 3.86469 |
| 5 | 35 | 0.65000 | 0.68889 | 1.31010 | 0.62222 | 3.19715 |
| 6 | 41 | 0.65000 | 0.80000 | 0.70550 | 0.71111 | 1.64283 |
| **7** | **137** | **0.82500** | **0.88889** | **0.58596** | **0.84444** | **0.92168** |
| 8 | 28 | 0.45000 | 0.66667 | 1.29982 | 0.51111 | 4.33535 |

Table 3.2: First network: results of the first training series with the CK+ database, for every fold.

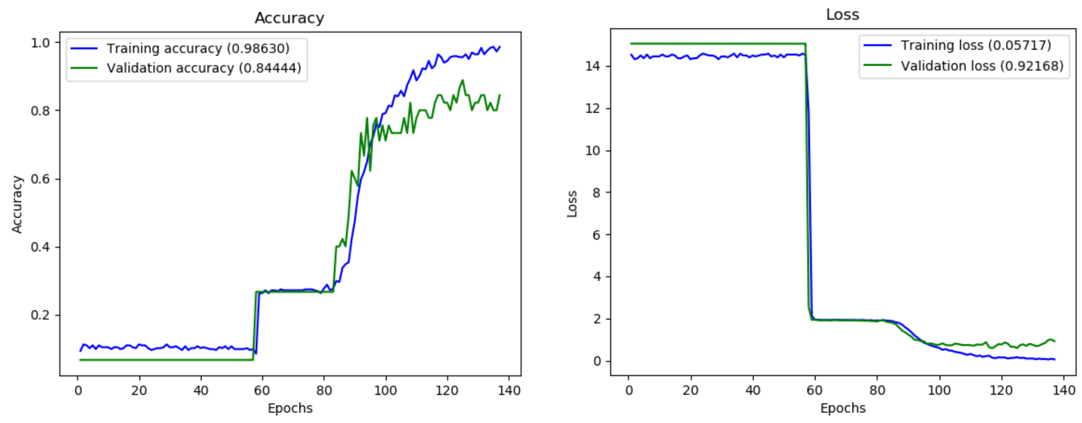


Figure 3.7: First network, CK+ database, first training series: loss and accuracy of the best network.

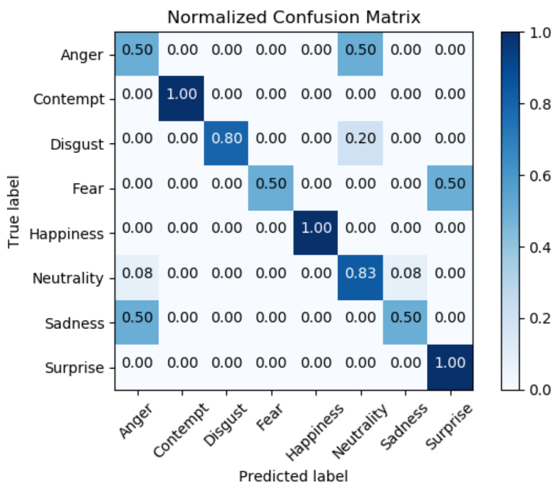


Figure 3.8: First network, CK+ database, first training series: normalized confusion matrix of the best network.

We can see that the results are surprisingly good, but the graphs are really bizarre (figure 3.7). From the loss graph in particular, but also from the accuracy ones, we can see that there is a big jump around epoch 60, indicating that the network has begun to learn something; around epoch 90, there is another rapid increase in the accuracy graph, of minor evidence in the loss one, which has brought the accuracy to its final value.

So, after this first encouraging training series, we ran another one, adding the scaling to a range normalization between 0 and 1, which, from now, we will call simply [0,1] normalization; we also added the data augmentation with the following parameters:

- a brightness range from 50% to 100%;
- horizontal flip enabled;
- rotation interval between +− 2.5 degrees;
- a shear range of +− 2.5%;
- a width and height shift range of +− 2.5%;
- a zoom transformation interval between +− 2.5%;

The results of the training are visible in table 3.3.

| Fold # | Epochs of Training | Test Accuracy | Max Validation Accuracy | Min Validation Loss | Final Validation Accuracy | Final Validation Loss |
|---|---|---|---|---|---|---|
| 0 | 101 | 0.87500 | 0.91304 | 0.24669 | 0.91304 | 0.41047 |
| 1 | 80 | 0.90000 | 0.86957 | 0.54744 | 0.78261 | 1.09485 |
| 2 | 89 | 0.77500 | 0.76087 | 1.03602 | 0.69565 | 1.05636 |
| 3 | 84 | 0.90000 | 0.80435 | 0.64164 | 0.73913 | 1.09686 |
| 4 | 23 | 0.30000 | 0.30435 | 1.89738 | 0.30435 | 1.90445 |
| 5 | 85 | 0.65000 | 0.66667 | 1.10012 | 0.62222 | 1.32859 |
| **6** | **115** | **0.92500** | **0.88889** | **0.32270** | **0.88889** | **0.45699** |
| 7 | 101 | 0.80000 | 0.84444 | 0.42916 | 0.80000 | 0.63079 |
| 8 | 113 | 0.87500 | 0.97778 | 0.06847 | 0.86667 | 0.81287 |

Table 3.3: First network: Results of the second training series with the CK+ database, for every fold.

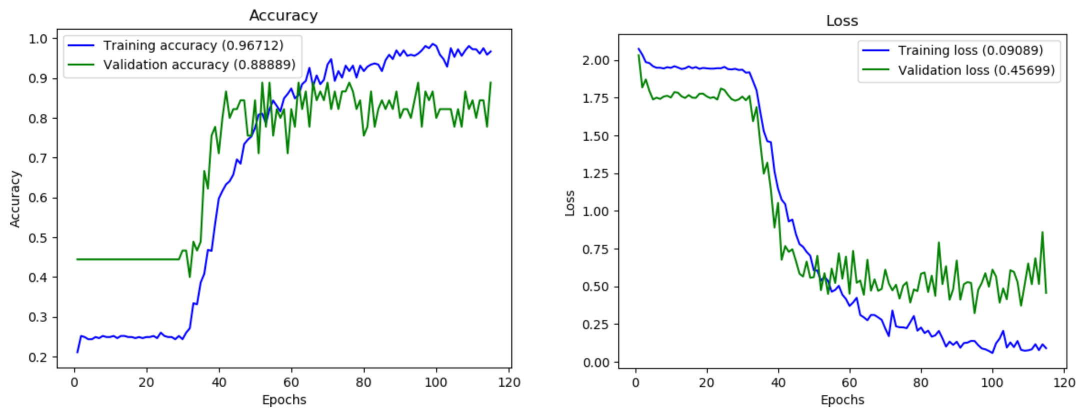
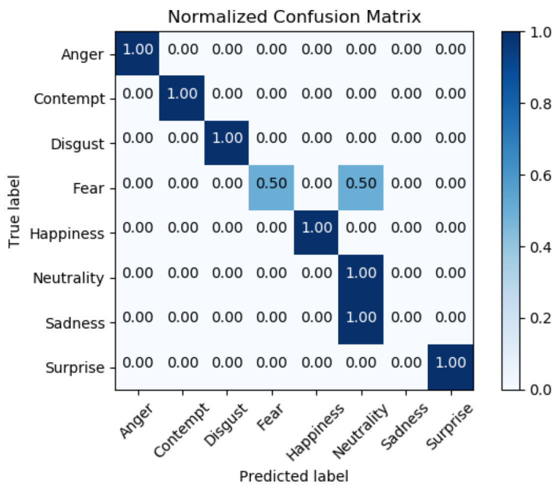


Figure 3.9: First network, CK+ database, second training series: loss and accuracy of the best network.



|  | Anger | Contempt | Disgust | Fear | Happiness | Neutrality | Sadness | Surprise |
|---|---|---|---|---|---|---|---|---|
| Anger | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Contempt | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Disgust | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Fear | 0.00 | 0.00 | 0.00 | 0.50 | 0.00 | 0.50 | 0.00 | 0.00 |
| Happiness | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 |
| Neutrality | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| Sadness | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| Surprise | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 |

Figure 3.10: First network, CK+ database, second training series: normalized confusion matrix of the best network.

We can see that the best network, achieved using the sixth fold as validation dataset, obtained a test accuracy of 92.5% after 115 epochs of training: a 10% improvement over the first test. We can see that even the graphs are slightly improved (figure 3.9). The bad thing is that the network has not learned well how to classify the sadness facial expression: the confusion matrix shows an accuracy of 0% for this state.

In our last training series with the CK+, we replaced the [0,1] normalization with the z-score standardization: the results are displayed in table 3.4.

| **Fold #** | **Epochs of Training** | **Test Accuracy** | **Max Validation Accuracy** | **Min Validation Loss** | **Final Validation Accuracy** | **Final Validation Loss** |
|---|---|---|---|---|---|---|
| 0 | 51 | 0.77500 | 0.80435 | 0.71713 | 0.76087 | 1.12720 |
| **1** | **46** | **0.92500** | **0.80435** | **0.54962** | **0.78261** | **0.55664** |
| 2 | 56 | 0.92500 | 0.82609 | 0.86837 | 0.71739 | 2.34957 |
| 3 | 21 | 0.30000 | 0.34783 | 1.98085 | 0.34783 | 2.06859 |
| 4 | 60 | 0.77500 | 0.71739 | 0.78750 | 0.67391 | 1.09214 |
| 5 | 48 | 0.77500 | 0.80000 | 0.85224 | 0.75556 | 1.48810 |
| 6 | 76 | 0.80000 | 0.91111 | 0.28022 | 0.82222 | 0.69303 |
| 7 | 60 | 0.82500 | 0.91111 | 0.31529 | 0.88889 | 0.51948 |
| 8 | 105 | 0.87500 | 0.97778 | 0.14559 | 0.88889 | 0.65755 |

Table 3.4: First network: Results of the third training series with the CK+ database, for every fold.

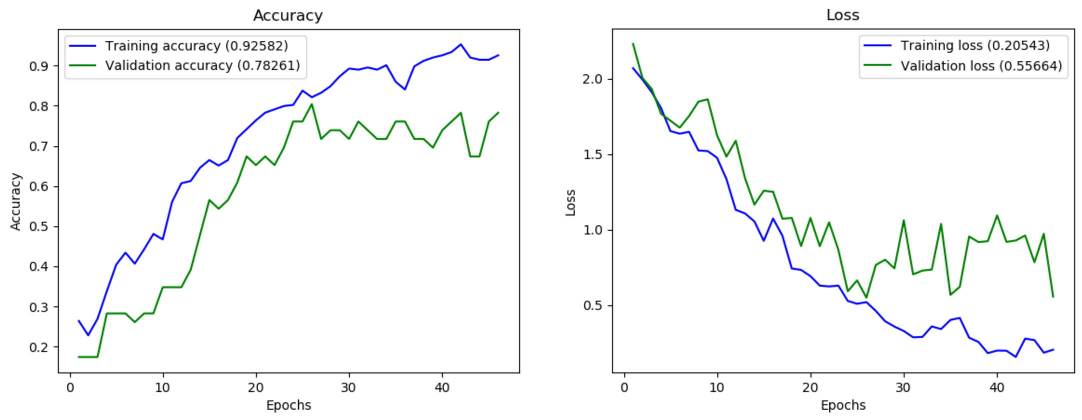


Figure 3.11: First network, CK+ database, third training series: loss and accuracy of the best network.

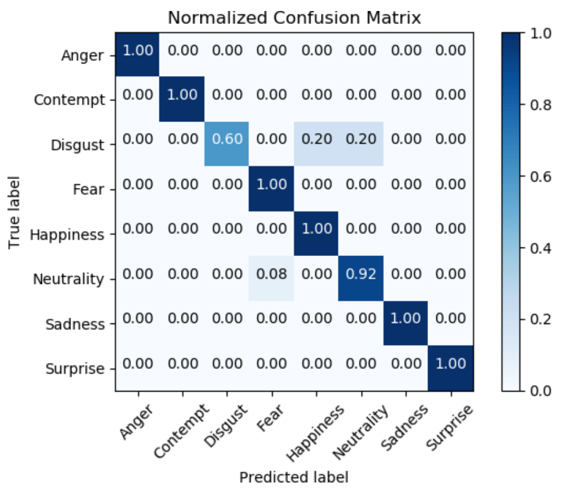


Figure 3.12: First network, CK+ database, third training series: normalized confusion matrix of the best network.

It can be seen that two networks achieved a 92.5% of test accuracy. If the validation losses are also observed, however, it is possible to note that the network obtained using the second fold as validation dataset, has a higher overfitting at the end of the training: therefore, the best neural network is the other. Compared to the best network obtained with the [0,1] normalization, this one had a faster convergence, but its descent has been disturbed in the same way: lowering the learning rate, we obtained cleaner graphs, but with a slower convergence.

In these training series, we obtained results in line with those stated in the paper in which this neural network was presented; now we will do new tests using a different database.

### FER2013 database

After doing some test with the CK+ database, we have adapted the network in order to work with the FER2013 database [18], which is a more populous database than CK+, but with very small images size: only 48x48 pixels. It is also important to add that the basic FER2013, compared to CK+, does not have the contempt state, so it has only 7 states. We used the same settings and hyperparameters that we used for the training made with the CK+ database, increasing only the batch size to 100. For FER2013, the subdivision between datasets, obtained with the same settings listed above, is as follows:

- 28712 photos for train;
- 3590 photos for validation;

- 3585 photos for test.

The results of the first training series are reported in table 3.5.

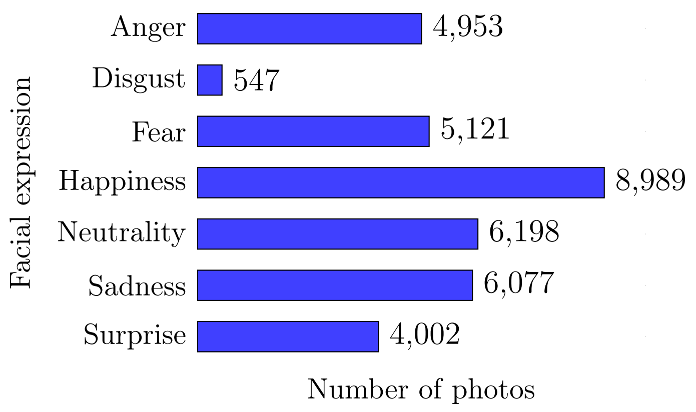


Figure 3.13: Subdivision of the images between classes in the FER2013 database.

| Fold # | Epochs of Training | Test Accuracy | Max Validation Accuracy | Min Validation Loss | Final Validation Accuracy | Final Validation Loss |
|---|---|---|---|---|---|---|
| 0 | 27 | 0.55593 | 0.57772 | 1.15456 | 0.56490 | 2.10535 |
| 1 | 27 | 0.55621 | 0.58735 | 1.14676 | 0.56701 | 2.01254 |
| 2 | 28 | 0.54700 | 0.56952 | 1.19026 | 0.55364 | 1.96380 |
| 3 | 28 | 0.55537 | 0.56172 | 1.21151 | 0.55057 | 2.44195 |
| 4 | 30 | 0.56569 | 0.56980 | 1.17097 | 0.56367 | 2.21624 |
| 5 | 29 | 0.55872 | 0.57398 | 1.19150 | 0.55893 | 2.04431 |
| **6** | **29** | **0.57127** | **0.58651** | **1.16405** | **0.57816** | **1.99641** |
| 7 | 29 | 0.54840 | 0.55949 | 1.20620 | 0.55336 | 2.01819 |
| 8 | 30 | 0.56179 | 0.58373 | 1.16308 | 0.57648 | 2.28562 |

Table 3.5: First network: Results of the first training series with the FER2013 database, for every fold.

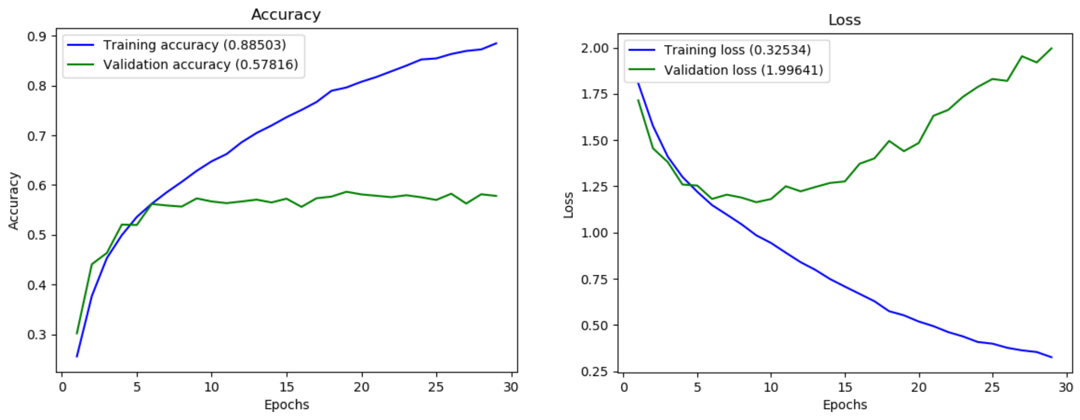


Figure 3.14: First network, FER2013 database, first training series: loss and accuracy of the best network.

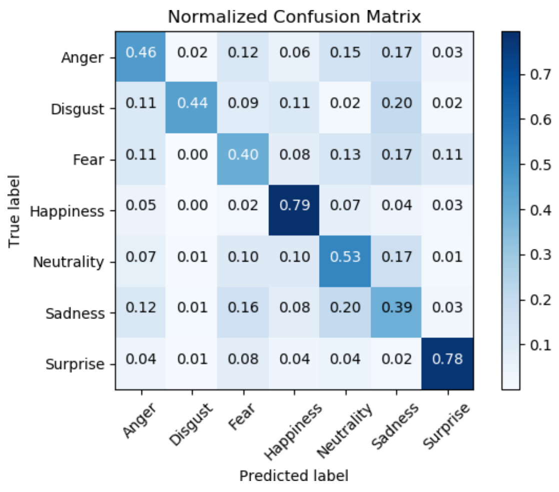


Figure 3.15: First network, FER2013 database, first training series: normalized confusion matrix of the best network.

We can see that the best network is the one obtained using the sixth fold as a validation dataset, which obtained a test accuracy of 57.127% after 29 epochs of training. From the final validation losses and from the loss graph of the best network (figure 3.14), it is clear that all the networks gone overfitting after a small number of epochs.

We want to reduce this effect: so, for the second training series, we added the [0,1] normalization and the data augmentation, with the same options that we used for the CK+. We can see the results in table 3.6.

| **Fold #** | **Epochs of Training** | **Test Accuracy** | **Max Validation Accuracy** | **Min Validation Loss** | **Final Validation Accuracy** | **Final Validation Loss** |
|---|---|---|---|---|---|---|
| 0 | 23 | 0.25049 | 0.25571 | 1.80701 | 0.25292 | 1.80844 |
| 1 | 30 | 0.25049 | 0.24464 | 1.81092 | 0.24464 | 1.81140 |
| 2 | 29 | 0.25049 | 0.25383 | 1.80251 | 0.25383 | 1.80385 |
| **3** | **80** | **0.61395** | **0.62413** | **1.03863** | **0.62413** | **1.06054** |
| 4 | 31 | 0.25049 | 0.24909 | 1.80709 | 0.24909 | 1.80802 |
| 5 | 69 | 0.25049 | 0.25383 | 1.80803 | 0.25383 | 1.80840 |
| 6 | 110 | 0.60223 | 0.61326 | 1.04642 | 0.61326 | 1.05739 |
| 7 | 59 | 0.25049 | 0.24659 | 1.81474 | 0.24659 | 1.81568 |
| 8 | 150 | 0.50265 | 0.50878 | 1.30024 | 0.49875 | 1.30781 |

Table 3.6: First network: Results of the second training series with the FER2013 database, for every fold.

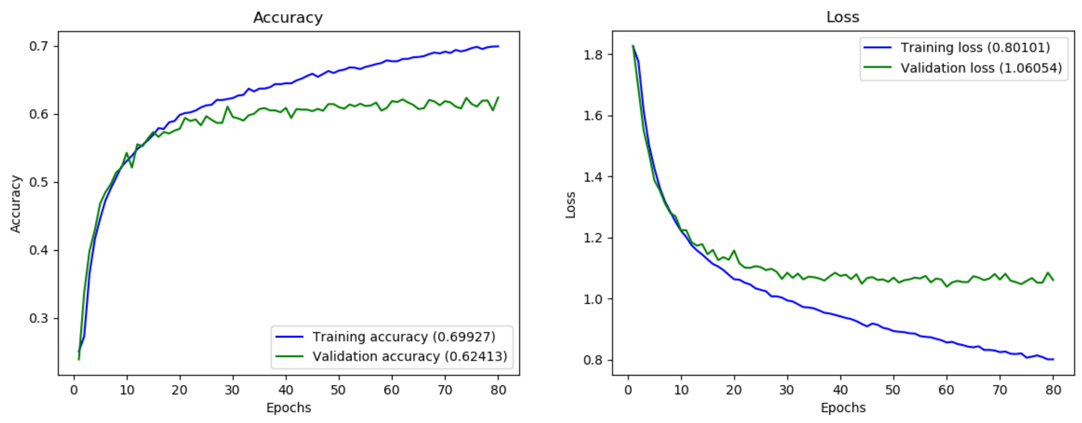
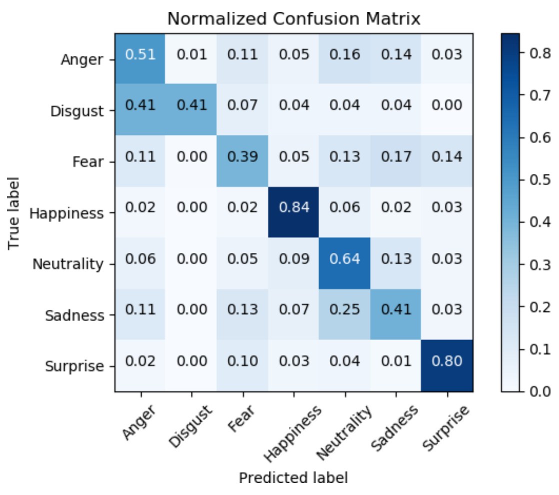


Figure 3.16: First network, FER2013 database, second training series: loss and accuracy of the best network.



| True label \ Predicted label | Anger | Disgust | Fear | Happiness | Neutrality | Sadness | Surprise |
|---|---|---|---|---|---|---|---|
| Anger | 0.51 | 0.01 | 0.11 | 0.05 | 0.16 | 0.14 | 0.03 |
| Disgust | 0.41 | 0.41 | 0.07 | 0.04 | 0.04 | 0.04 | 0.00 |
| Fear | 0.11 | 0.00 | 0.39 | 0.05 | 0.13 | 0.17 | 0.14 |
| Happiness | 0.02 | 0.00 | 0.02 | 0.84 | 0.06 | 0.02 | 0.03 |
| Neutrality | 0.06 | 0.00 | 0.05 | 0.09 | 0.64 | 0.13 | 0.03 |
| Sadness | 0.11 | 0.00 | 0.13 | 0.07 | 0.25 | 0.41 | 0.03 |
| Surprise | 0.02 | 0.00 | 0.10 | 0.03 | 0.04 | 0.01 | 0.80 |

Figure 3.17: First network, FER2013 database, second training series: normalized confusion matrix of the best network.

With these settings, we can see that most of the networks have not been able to converge, but the best network achieved a test accuracy of 61.395% after 80 epochs of training: not an incredible percentage, but definitely improvable. We can also note a clear improvement in the form of the graphs, which now tends to saturate in the last epochs (figure 3.16). Replacing the [0,1] normalization with the z-score standardization, we got the results displayed in table 3.7.

| Fold # | Epochs of Training | Test Accuracy | Max Validation Accuracy | Min Validation Loss | Final Validation Accuracy | Final Validation Loss |
|---|---|---|---|---|---|---|
| 0 | 63 | 0.60307 | 0.62423 | 1.07525 | 0.61281 | 1.11469 |
| **1** | **57** | **0.62455** | **0.63360** | **1.02003** | **0.62998** | **1.06835** |
| 2 | 51 | 0.61506 | 0.62469 | 1.04563 | 0.61744 | 1.09491 |
| 3 | 49 | 0.61646 | 0.62887 | 1.05984 | 0.61884 | 1.15689 |
| 4 | 56 | 0.60363 | 0.61688 | 1.07400 | 0.60463 | 1.14892 |
| 5 | 56 | 0.61925 | 0.63054 | 1.01599 | 0.61856 | 1.07072 |
| 6 | 56 | 0.60753 | 0.61744 | 1.05614 | 0.60853 | 1.09714 |
| 7 | 50 | 0.60223 | 0.61521 | 1.08188 | 0.60184 | 1.13446 |
| 8 | 62 | 0.60335 | 0.61772 | 1.07363 | 0.60853 | 1.17247 |

Table 3.7: First network: Results of the third training series with the FER2013 database, for every fold.

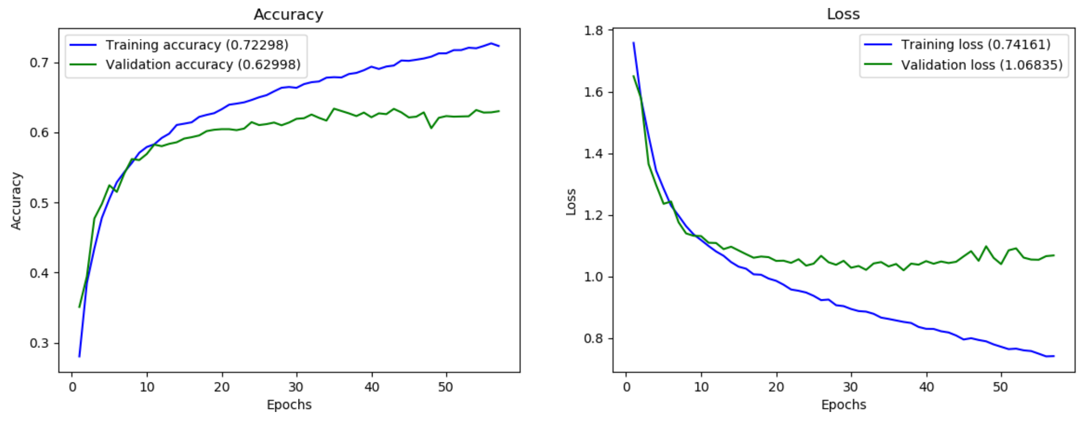
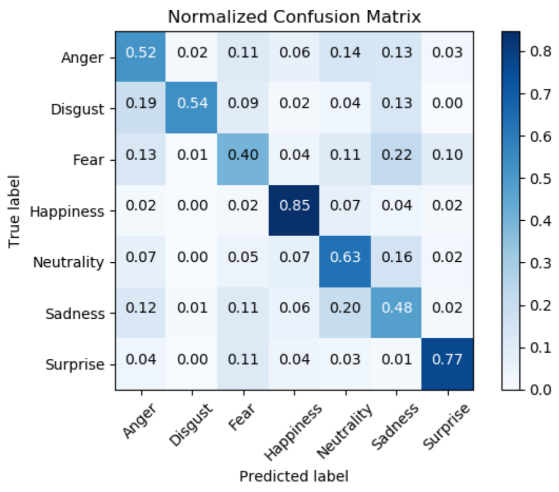


Figure 3.18: First network, FER2013 database, third training series: loss and accuracy of the best network.



| True label | Anger | Disgust | Fear | Happiness | Neutrality | Sadness | Surprise |
|---|---|---|---|---|---|---|---|
| Anger | 0.52 | 0.02 | 0.11 | 0.06 | 0.14 | 0.13 | 0.03 |
| Disgust | 0.19 | 0.54 | 0.09 | 0.02 | 0.04 | 0.13 | 0.00 |
| Fear | 0.13 | 0.01 | 0.40 | 0.04 | 0.11 | 0.22 | 0.10 |
| Happiness | 0.02 | 0.00 | 0.02 | 0.85 | 0.07 | 0.04 | 0.02 |
| Neutrality | 0.07 | 0.00 | 0.05 | 0.07 | 0.63 | 0.16 | 0.02 |
| Sadness | 0.12 | 0.01 | 0.11 | 0.06 | 0.20 | 0.48 | 0.02 |
| Surprise | 0.04 | 0.00 | 0.11 | 0.04 | 0.03 | 0.01 | 0.77 |

Figure 3.19: First network, FER2013 database, third training series: normalized confusion matrix of the best network.

We can see that the best network reached a test accuracy of 62.455% after 57 epochs of training: a small improvement, but far from the test accuracy of about 69% that others have achieved with the same database [27].

We will now use the FER2013 with the FER+ annotations, that relabels FEDC in a more accurate way and allows to remove some images that do not contain human faces. Differently from FER2013, it has also the contempt state, for a total of 8 facial expressions (figure 3.20).

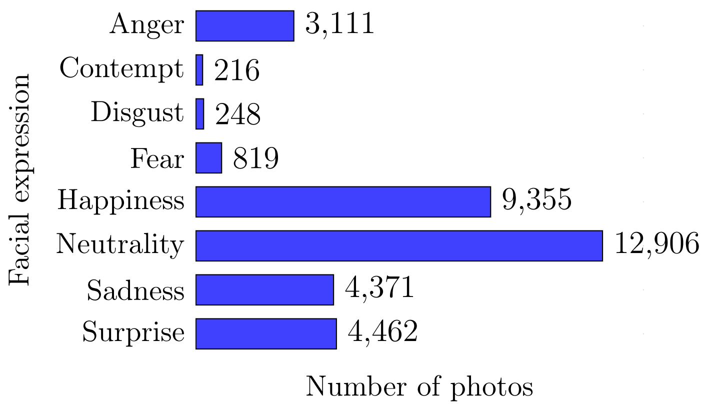


Figure 3.20: Subdivision of the images between classes in the FER2013 database with the FER+ annotations.

With the FER2013 database with the FER+ annotations, we got the results displayed in table 3.8.

| **Fold #** | **Epochs of Training** | **Test Accuracy** | **Max Validation Accuracy** | **Min Validation Loss** | **Final Validation Accuracy** | **Final Validation Loss** |
|---|---|---|---|---|---|---|
| 0 | 52 | 0.76784 | 0.76873 | 0.70279 | 0.75775 | 0.74301 |
| 1 | 45 | 0.78138 | 0.77296 | 0.68491 | 0.77014 | 0.73256 |
| 2 | 48 | 0.75515 | 0.76219 | 0.73907 | 0.75148 | 0.81934 |
| 3 | 50 | 0.77207 | 0.78670 | 0.65363 | 0.77937 | 0.69809 |
| 4 | 44 | 0.76869 | 0.79121 | 0.63467 | 0.77318 | 0.67572 |
| 5 | 50 | 0.77123 | 0.77966 | 0.69770 | 0.77120 | 0.72551 |
| **6** | **49** | **0.78477** | **0.78980** | **0.65247** | **0.78980** | **0.68151** |
| 7 | 45 | 0.76756 | 0.76698 | 0.71291 | 0.75768 | 0.73650 |
| 8 | 47 | 0.78364 | 0.78670 | 0.65710 | 0.77543 | 0.73530 |

Table 3.8: First network: Results of the fourth training series with the FER2013 database with the FER+ annotation, for every fold.

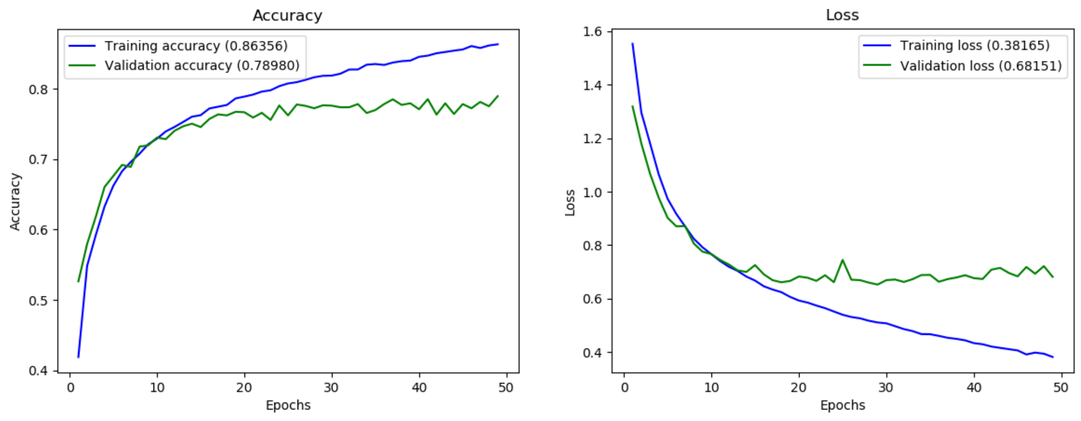
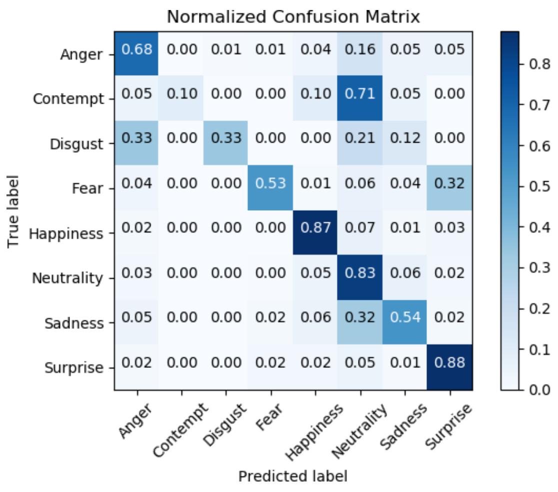


Figure 3.21: First network, FER2013 database with the FER+ annotation, fourth training series: loss and accuracy of the best network.



| | Anger | Contempt | Disgust | Fear | Happiness | Neutrality | Sadness | Surprise |
|---|---|---|---|---|---|---|---|---|
| Anger | 0.68 | 0.00 | 0.01 | 0.01 | 0.04 | 0.16 | 0.05 | 0.05 |
| Contempt | 0.05 | 0.10 | 0.00 | 0.00 | 0.10 | 0.71 | 0.05 | 0.00 |
| Disgust | 0.33 | 0.00 | 0.33 | 0.00 | 0.00 | 0.21 | 0.12 | 0.00 |
| Fear | 0.04 | 0.00 | 0.00 | 0.53 | 0.01 | 0.06 | 0.04 | 0.32 |
| Happiness | 0.02 | 0.00 | 0.00 | 0.00 | 0.87 | 0.07 | 0.01 | 0.03 |
| Neutrality | 0.03 | 0.00 | 0.00 | 0.00 | 0.05 | 0.83 | 0.06 | 0.02 |
| Sadness | 0.05 | 0.00 | 0.00 | 0.02 | 0.06 | 0.32 | 0.54 | 0.02 |
| Surprise | 0.02 | 0.00 | 0.00 | 0.02 | 0.02 | 0.05 | 0.01 | 0.88 |

Figure 3.22: First network, FER2013 database with the FER+ annotation, fourth training series: normalized confusion matrix of the best network.

Using the FER+ annotations, the test accuracies have significantly improved: the best has reached a value of 78.477%. However, this value is about 8% lower than that obtained by other researchers [8], so it can be certainly improved. Finally, it is important to note the poor accuracy, of the neural network, with expressions such as contempt and disgust (figure 3.21), probably caused by the very low number of images present for these two states.

### Database Ensemble of all the facial expressions databases supported by FEDC

We have already talked, in the second chapter, about the possible advantages of creating a **Database Ensemble**; in particular, this can be of great help for our task, because we want to create a neural network that is able to generalize well and that can work even with lights coming from different directions: a common situation in the passengers compartment of a car. So, we created a big Database Ensemble merging the databases supported by FEDC, and choosing the following options:

- width and height of 48x48 pixels, in order to adapt the images size of the other databases to those of FER2013;
- grayscale option to true;
- face detection and crop option to true.

For the FER2013 database, we chose to use the FER+ annotations, which previously significantly increased the accuracy of the test.

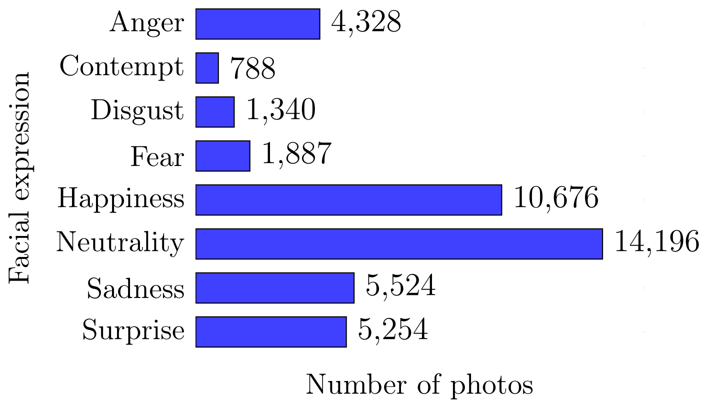


Figure 3.23: Subdivision of the images between classes in the Database Ensemble of all the facial expressions databases supported by FEDC, created merging the CK+, the FACES, the FER2013 with the FER+ annotations, the JAFFE, the MUG, the RaFD and the SFEW 2.0 databases.

For this Database Ensemble, the subdivision between datasets, obtained using the same settings as usual, is as follows:

- 35212 photos for train;
- 4402 photos for validation;
- 4379 photos for test.

In the first train, we obtained the result reported in table 3.9.

| Fold # | Epochs of Training | Test Accuracy | Max Validation Accuracy | Min Validation Loss | Final Validation Accuracy | Final Validation Loss |
|---|---|---|---|---|---|---|
| 0 | 27 | 0.73738 | 0.75170 | 0.75490 | 0.74739 | 1.22234 |
| 1 | 25 | 0.71615 | 0.72717 | 0.86230 | 0.71059 | 1.63357 |
| 2 | 27 | 0.74081 | 0.74057 | 0.81986 | 0.73308 | 1.37405 |
| 3 | 30 | 0.74606 | 0.76261 | 0.75066 | 0.75557 | 1.30018 |
| 4 | 27 | 0.72551 | 0.73603 | 0.81603 | 0.73171 | 1.35684 |
| 5 | 27 | 0.68235 | 0.68780 | 0.96217 | 0.68530 | 1.86440 |
| **6** | **28** | **0.74789** | **0.75778** | **0.79279** | **0.75778** | **1.35676** |
| 7 | 26 | 0.73578 | 0.74688 | 0.78196 | 0.74688 | 1.27023 |
| 8 | 28 | 0.74195 | 0.74960 | 0.79738 | 0.74960 | 1.32298 |

Table 3.9: First network: Results of the first training series with the Database Ensemble of all the facial expressions databases, for every fold.

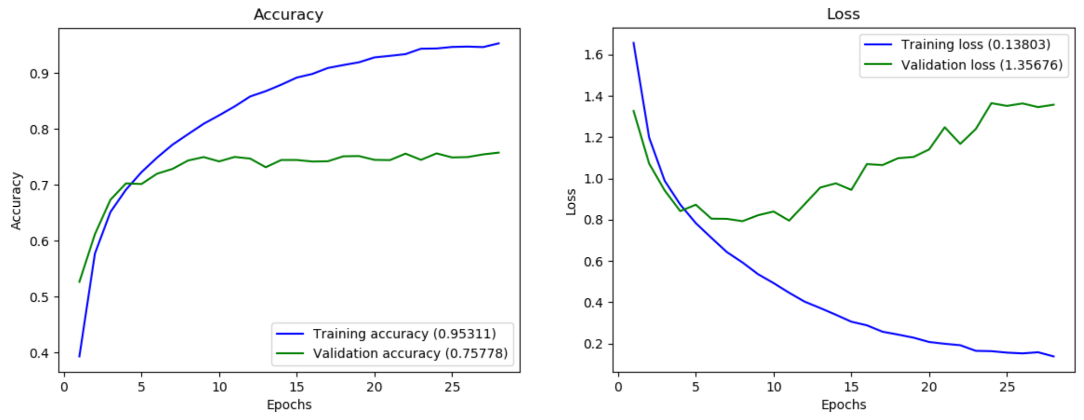
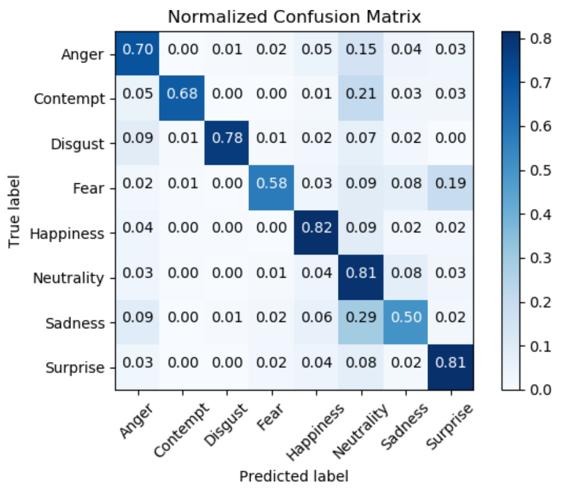


Figure 3.24: First network, Database Ensemble of all the facial expressions databases, first training series: loss and accuracy of the best network.



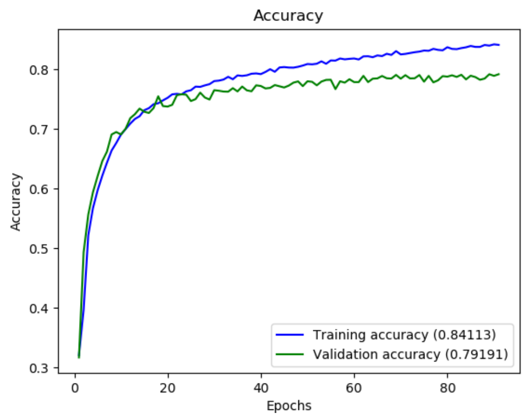
| True label | Anger | Contempt | Disgust | Fear | Happiness | Neutrality | Sadness | Surprise |
|---|---|---|---|---|---|---|---|---|
| Anger | 0.70 | 0.00 | 0.01 | 0.02 | 0.05 | 0.15 | 0.04 | 0.03 |
| Contempt | 0.05 | 0.68 | 0.00 | 0.00 | 0.01 | 0.21 | 0.03 | 0.03 |
| Disgust | 0.09 | 0.01 | 0.78 | 0.01 | 0.02 | 0.07 | 0.02 | 0.00 |
| Fear | 0.02 | 0.01 | 0.00 | 0.58 | 0.03 | 0.09 | 0.08 | 0.19 |
| Happiness | 0.04 | 0.00 | 0.00 | 0.00 | 0.82 | 0.09 | 0.02 | 0.02 |
| Neutrality | 0.03 | 0.00 | 0.00 | 0.01 | 0.04 | 0.81 | 0.08 | 0.03 |
| Sadness | 0.09 | 0.00 | 0.01 | 0.02 | 0.06 | 0.29 | 0.50 | 0.02 |
| Surprise | 0.03 | 0.00 | 0.00 | 0.02 | 0.04 | 0.08 | 0.02 | 0.81 |

Figure 3.25: First network, Database Ensemble of all the facial expressions databases, first training series: normalized confusion matrix of the best network.

The best neural network, trained using the sixth fold as validation dataset, obtained a test accuracy of 74.789% after only 28 epochs: not bad, considering that we are not using neither data augmentation nor normalization. However, the presence of overfitting is evident (figure 3.24), therefore we have tried to reduce it using the [0,1] normalization and the data augmentation; the results can be seen in table 3.10.

| Fold # | Epochs of Training | Test Accuracy | Max Validation Accuracy | Min Validation Loss | Final Validation Accuracy | Final Validation Loss |
|---|---|---|---|---|---|---|
| **0** | **91** | **0.78854** | **0.79191** | **0.60639** | **0.79191** | **0.62411** |
| 1 | 98 | 0.78625 | 0.79123 | 0.63274 | 0.79123 | 0.63724 |
| 2 | 135 | 0.75634 | 0.76761 | 0.68421 | 0.75738 | 0.71353 |
| 3 | 121 | 0.76296 | 0.78782 | 0.64510 | 0.77715 | 0.67840 |
| 4 | 91 | 0.74127 | 0.76056 | 0.71522 | 0.74716 | 0.72823 |
| 5 | 102 | 0.78671 | 0.80345 | 0.59054 | 0.79096 | 0.63663 |
| 6 | 61 | 0.78420 | 0.78709 | 0.62844 | 0.78050 | 0.65949 |
| 7 | 119 | 0.75063 | 0.75596 | 0.71076 | 0.74847 | 0.72175 |
| 8 | 97 | 0.78511 | 0.78823 | 0.61428 | 0.78028 | 0.65644 |

Table 3.10: First network: Results of the second training series with the Database Ensemble of all the facial expressions databases, for every fold.



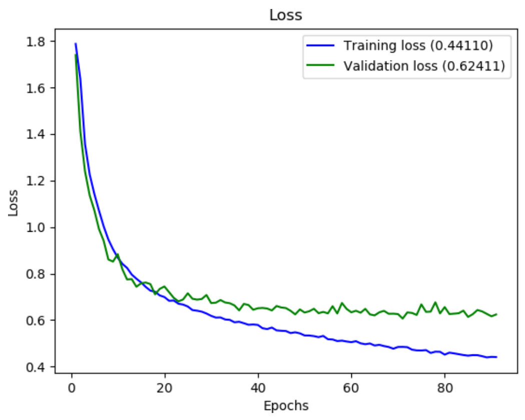


Figure 3.26: First network, Database Ensemble of all the facial expressions databases, second training series: loss and accuracy of the best network.

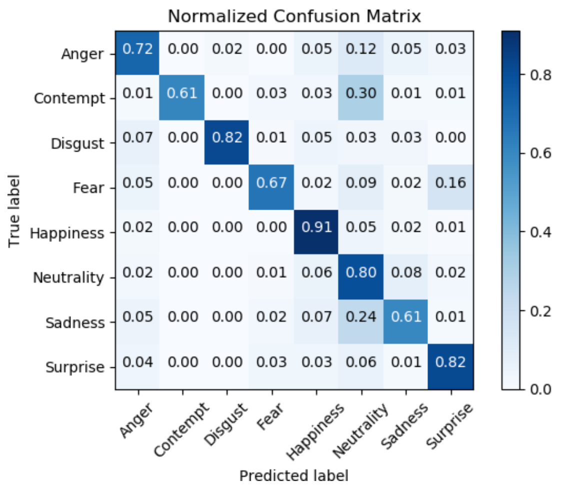


Figure 3.27: First network, Database Ensemble of all the facial expressions databases, second training series: normalized confusion matrix of the best network.

In this second training series, the results are clearly better than before, with less overfitting (figure 3.26) and with a test accuracy that reached 78.854%. We then performed a last training series, replacing the [0,1] normalization with the z-score standardization; the results can be seen in table 3.11.

| **Fold #** | **Epochs of Training** | **Test Accuracy** | **Max Validation Accuracy** | **Min Validation Loss** | **Final Validation Accuracy** | **Final Validation Loss** |
|---|---|---|---|---|---|---|
| 0 | 55 | 0.78397 | 0.79464 | 0.62088 | 0.78305 | 0.67163 |
| 1 | 80 | 0.79059 | 0.79032 | 0.63431 | 0.78487 | 0.66214 |
| 2 | 61 | 0.79767 | 0.79487 | 0.62683 | 0.79259 | 0.74198 |
| **3** | **96** | **0.80384** | **0.81508** | **0.56137** | **0.81418** | **0.60667** |
| 4 | 48 | 0.79447 | 0.79782 | 0.64320 | 0.78919 | 0.66748 |
| 5 | 76 | 0.79447 | 0.80164 | 0.59570 | 0.79936 | 0.63107 |
| 6 | 55 | 0.78785 | 0.78959 | 0.66027 | 0.78687 | 0.70305 |
| 7 | 59 | 0.79013 | 0.79232 | 0.62626 | 0.78391 | 0.67526 |
| 8 | 62 | 0.79744 | 0.79505 | 0.61571 | 0.78664 | 0.64963 |

Table 3.11: First network: Results of the third training series with the Database Ensemble of all the facial expressions databases, for every fold.

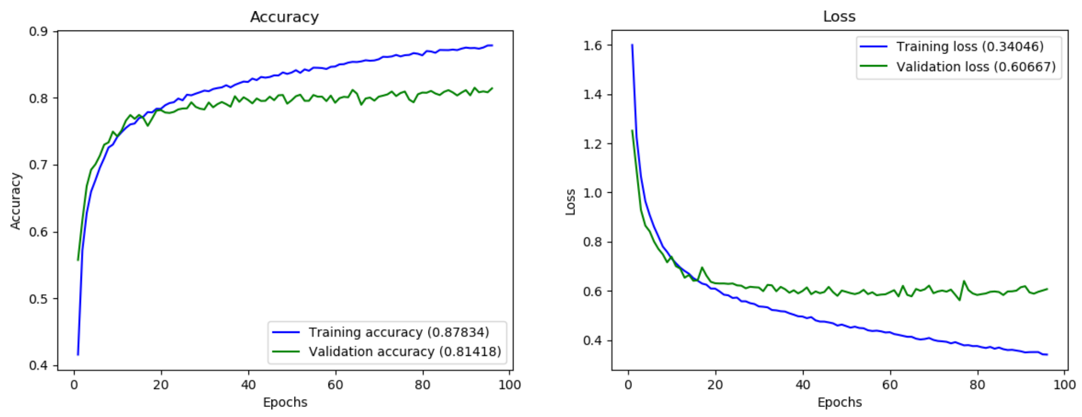


Figure 3.28: First network, Database Ensemble of all the facial expressions databases, third training series: loss and accuracy of the best network.

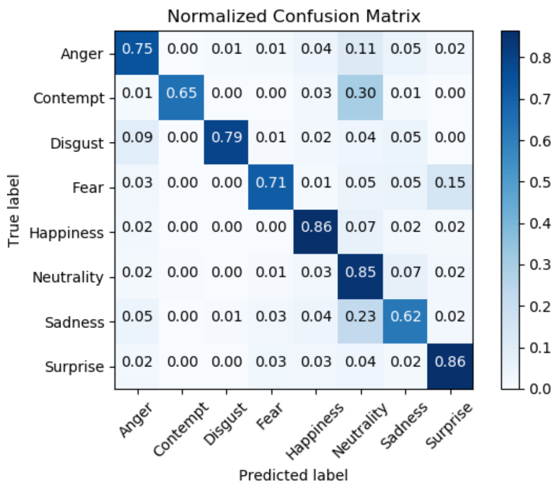


Figure 3.29: First network, Database Ensemble of all the facial expressions databases, third training series: normalized confusion matrix of the best network.

The neural network that obtained the highest test accuracy is the one having the third fold as validation dataset: it was capable of reaching an accuracy of 80.384%, a good result. We can also see that the network is able to recognize all facial expressions well enough (figure 3.29).

### Database Ensemble of all the posed facial expressions databases supported by FEDC

As last test with this network, we decided to create a Database Ensemble made only with the posed facial expressions databases that are supported by FEDC [48], which are, in order, CK+ database [20] [24], FACES database [15], JAFFE database [25], MUG database [7], and RaFD database [22]. Our intention is to increase the test accuracy: using only posed facial expressions, we expect to improve the result obtained with the previous Database Ensemble. So, we created this database by combining the five mentioned above, preparing the images with FEDC and setting the following options:

- width and a height of 120x120: we can use the standard images resolution of the neural network because we will not use the FER2013 database;
- grayscale option to true;
- face detection and crop option to true.

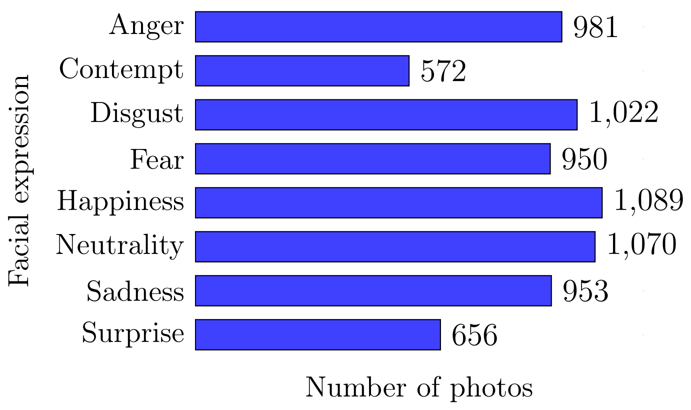


Figure 3.30: Subdivision of the images between classes in the Database Ensemble of posed facial expressions, created merging the CK+, the FACES, the JAFFE, the MUG and the RaFD databases.

For this Database Ensemble, the subdivision between datasets, obtained with the same settings listed above, is as follows:

- 5847 photos for train;
- 731 photos for validation;
- 715 photos for test.

As usual until now, we started the first training series using the default settings.

| Fold # | Epochs of Training | Test Accuracy | Max Validation Accuracy | Min Validation Loss | Final Validation Accuracy | Final Validation Loss |
|---|---|---|---|---|---|---|
| 0 | 46 | 0.92587 | 0.93707 | 0.30826 | 0.93434 | 0.38069 |
| 1 | 42 | 0.93287 | 0.92202 | 0.28003 | 0.91655 | 0.40912 |
| 2 | 48 | 0.94266 | 0.91929 | 0.35101 | 0.91108 | 0.44459 |
| 3 | 46 | 0.92727 | 0.93160 | 0.31080 | 0.90014 | 0.47682 |
| **4** | **40** | **0.94685** | **0.90971** | **0.37273** | **0.90971** | **0.46919** |
| 5 | 50 | 0.92867 | 0.92066 | 0.35045 | 0.90424 | 0.45716 |
| 6 | 49 | 0.94406 | 0.91655 | 0.36122 | 0.91245 | 0.44036 |
| 7 | 39 | 0.93846 | 0.91382 | 0.41681 | 0.90698 | 0.45320 |
| 8 | 36 | 0.93566 | 0.91370 | 0.35453 | 0.90548 | 0.44373 |

Table 3.12: First network: Results of the first training series with the Database Ensemble of posed facial expressions, for every fold.

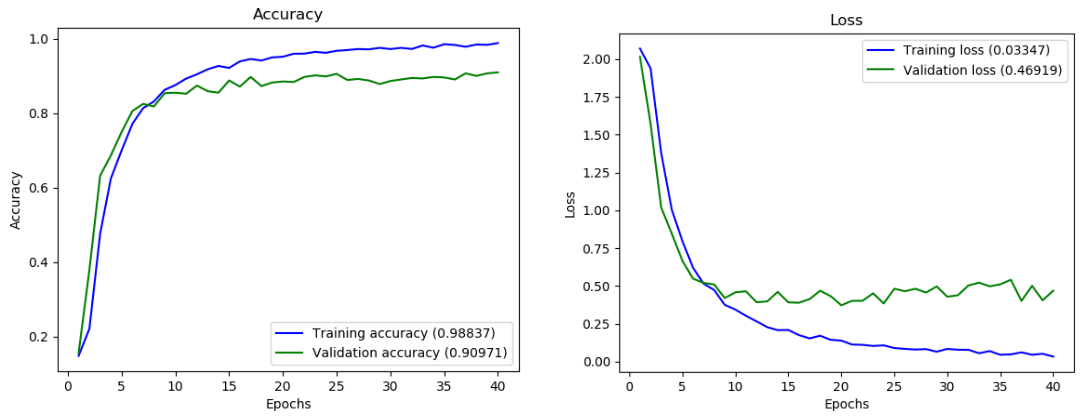
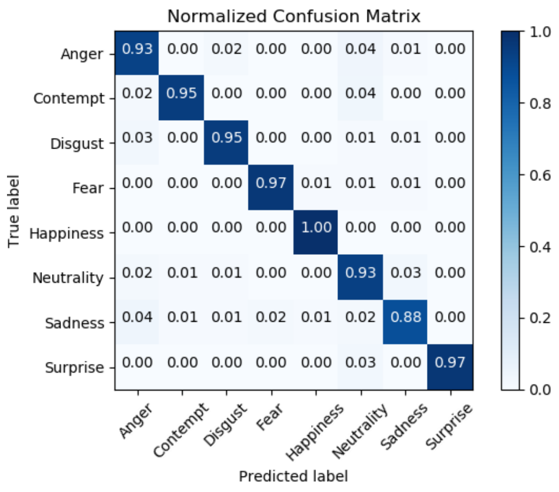


Figure 3.31: First network, Database Ensemble of posed facial expressions, first training series: loss and accuracy of the best network.



| True label | Anger | Contempt | Disgust | Fear | Happiness | Neutrality | Sadness | Surprise |
|---|---|---|---|---|---|---|---|---|
| Anger | 0.93 | 0.00 | 0.02 | 0.00 | 0.00 | 0.04 | 0.01 | 0.00 |
| Contempt | 0.02 | 0.95 | 0.00 | 0.00 | 0.00 | 0.04 | 0.00 | 0.00 |
| Disgust | 0.03 | 0.00 | 0.95 | 0.00 | 0.00 | 0.01 | 0.01 | 0.00 |
| Fear | 0.00 | 0.00 | 0.00 | 0.97 | 0.01 | 0.01 | 0.01 | 0.00 |
| Happiness | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 |
| Neutrality | 0.02 | 0.01 | 0.01 | 0.00 | 0.00 | 0.93 | 0.03 | 0.00 |
| Sadness | 0.04 | 0.01 | 0.01 | 0.02 | 0.01 | 0.02 | 0.88 | 0.00 |
| Surprise | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.03 | 0.00 | 0.97 |

Figure 3.32: First network, Database Ensemble of posed facial expressions, first training series: normalized confusion matrix of the best network.

We can see, from the table 3.12, that the results are, in general, very good, because all the neural networks have converged and have very similar values. The best network achieved an accuracy of 94.685%, only 2% better than those that went worse, confirming what was said previously. Now, we will make another training series adding the data augmentation, for simplicity always with the same values, and the [0,1] normalization.

| **Fold #** | **Epochs of Training** | **Test Accuracy** | **Max Validation Accuracy** | **Min Validation Loss** | **Final Validation Accuracy** | **Final Validation Loss** |
|---|---|---|---|---|---|---|
| 0 | 150 | 0.97063 | 0.96717 | 0.11521 | 0.95486 | 0.13700 |
| **1** | **137** | **0.97203** | **0.96990** | **0.14160** | **0.95896** | **0.15575** |
| 2 | 199 | 0.95944 | 0.93981 | 0.20904 | 0.92066 | 0.26187 |
| 3 | 66 | 0.96783 | 0.94938 | 0.16656 | 0.94938 | 0.16662 |
| 4 | 101 | 0.89790 | 0.87962 | 0.39859 | 0.85773 | 0.45697 |
| 5 | 50 | 0.14965 | 0.16005 | 2.05512 | 0.13817 | 2.05548 |
| 6 | 102 | 0.81958 | 0.80848 | 0.58044 | 0.80027 | 0.58339 |
| 7 | 24 | 0.14825 | 0.18057 | 2.06165 | 0.12722 | 2.06284 |
| 8 | 98 | 0.88252 | 0.84110 | 0.46822 | 0.83014 | 0.47432 |

Table 3.13: First network: Results of the second training series with the Database Ensemble of posed facial expressions, for every fold.

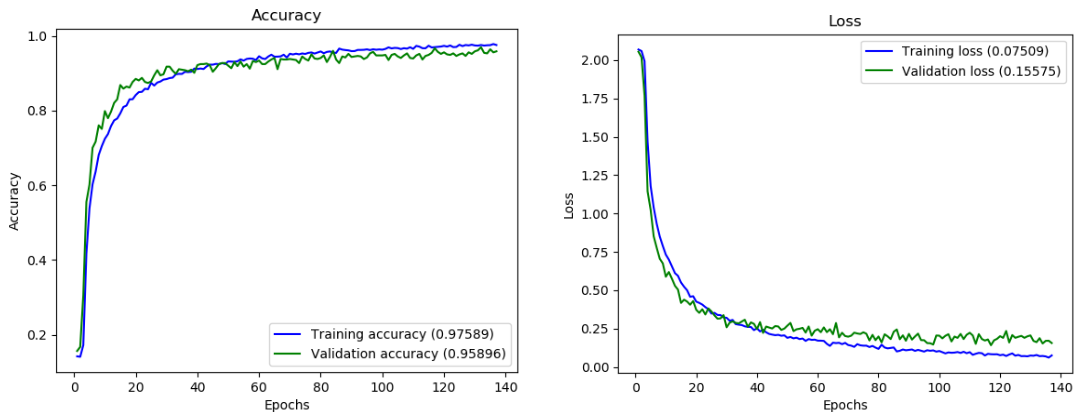


Figure 3.33: First network, Database Ensemble of posed facial expressions, second training series: loss and accuracy of the best network.

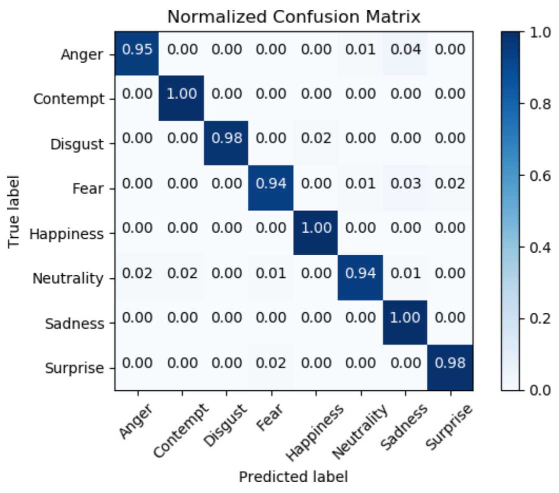


Figure 3.34: First network, Database Ensemble of posed facial expressions, second training series: normalized confusion matrix of the best network.

Respect to the first training series, from the table 3.13 it can be seen that not all neural networks have been able to converge, but the best one, obtained using the first fold as validation dataset, was able to achieve an accuracy of 97.203%. It has also a greater validation accuracy and a lower validation loss (figure 3.33), at the price of a higher training time: 137 epochs. As last test for this network, we replaced the [0,1] normalization with the z-score standardization, obtaining the results displayed in table 3.14.

| **Fold** # | **Epochs of Training** | **Test Accuracy** | **Max Validation Accuracy** | **Min Validation Loss** | **Final Validation Accuracy** | **Final Validation Loss** |
|---|---|---|---|---|---|---|
| 0 | 88 | 0.96783 | 0.95075 | 0.18734 | 0.94938 | 0.23673 |
| 1 | 80 | 0.96783 | 0.96170 | 0.15437 | 0.96170 | 0.19863 |
| 2 | 88 | 0.91469 | 0.90424 | 0.33568 | 0.88235 | 0.43428 |
| 3 | 74 | 0.94685 | 0.95075 | 0.20459 | 0.94802 | 0.25145 |
| 4 | 50 | 0.94406 | 0.91245 | 0.32322 | 0.91108 | 0.39697 |
| **5** | **61** | **0.97343** | **0.94802** | **0.16156** | **0.93570** | **0.19506** |
| 6 | 81 | 0.92587 | 0.89877 | 0.32843 | 0.88646 | 0.36951 |
| 7 | 74 | 0.96923 | 0.95349 | 0.21463 | 0.94528 | 0.24640 |
| 8 | 62 | 0.97203 | 0.94932 | 0.19354 | 0.93699 | 0.21231 |

Table 3.14: First network: Results of the third training series with the Database Ensemble of posed facial expressions, for every fold.

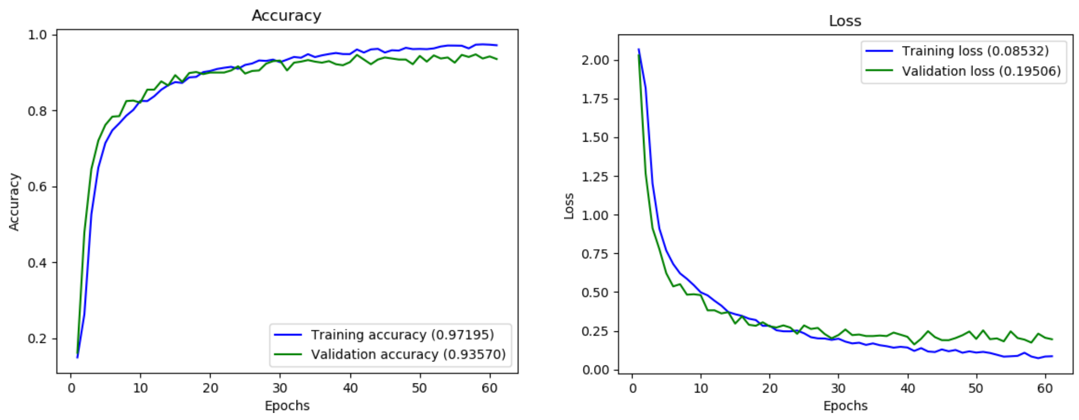
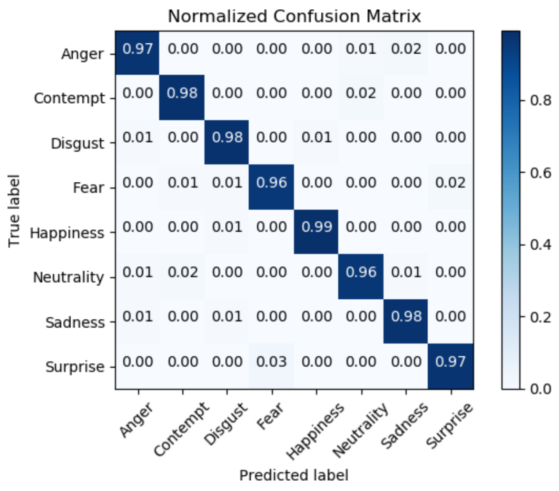


Figure 3.35: First network, Database Ensemble of posed facial expressions, third training series: loss and accuracy of the best network.



| | Anger | Contempt | Disgust | Fear | Happiness | Neutrality | Sadness | Surprise |
|---|---|---|---|---|---|---|---|---|
| Anger | 0.97 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.02 | 0.00 |
| Contempt | 0.00 | 0.98 | 0.00 | 0.00 | 0.00 | 0.02 | 0.00 | 0.00 |
| Disgust | 0.01 | 0.00 | 0.98 | 0.00 | 0.01 | 0.00 | 0.00 | 0.00 |
| Fear | 0.00 | 0.01 | 0.01 | 0.96 | 0.00 | 0.00 | 0.00 | 0.02 |
| Happiness | 0.00 | 0.00 | 0.01 | 0.00 | 0.99 | 0.00 | 0.00 | 0.00 |
| Neutrality | 0.01 | 0.02 | 0.00 | 0.00 | 0.00 | 0.96 | 0.01 | 0.00 |
| Sadness | 0.01 | 0.00 | 0.01 | 0.00 | 0.00 | 0.00 | 0.98 | 0.00 |
| Surprise | 0.00 | 0.00 | 0.00 | 0.03 | 0.00 | 0.00 | 0.00 | 0.97 |

Figure 3.36: First network, Database Ensemble of posed facial expressions, third training series: normalized confusion matrix of the best network.

These results are similar to those obtained with the second training series: all the network have converged, and the best performed slightly better, scoring a 97.343% as test accuracy. We succeeded in our intent: with this database, we have increased the test accuracy, respect to the previous Database Ensemble, by around 17%.

### 3.7.2 Neural network from the "Recognizing Facial Expressions Using a Shallow Convolutional Neural Network" paper

As previously said, we have chosen as second neural network the one described in the **Recognizing Facial Expressions Using a Shallow Convolutional Neural Network** paper [27]; so, as we did before for the other network, we implemented it in Keras [44] and we made some tests to evaluate its performance.

For each test, we used the EarlyStopping callback, set up to block training if, after 20 epochs, there were no improvements to the validation loss, and the ReduceLROnPlateau callback set to multiply the learning rate by 0.95 if, after 5 epochs, there were no improvements on the validation loss.

The initial configuration for our tests, with little modifications about the hyperparameters respect the ones reported in the paper, and to those used for the networks previously trained, was the following:

- batch size: 100;
- maximum number of epochs: 1000;
- learning rate: 0.0001;
- preliminary subdivision of 90%-10% between train and test set, made with FEDC [48];
- a 9-fold, which further divides the train dataset into variable train and validation datasets.

| **Layer #** | **Layer (type)** | **Output shape** | **Connected to** |
|---|---|---|---|
| 1 | input_1(InputLayer) | (1, 48, 48) | - |
| 2 | conv2d_1(Conv2D) | (44, 48, 48) | input_1 |
| 3 | max_pool2d_1(MaxPooling2D) | (44, 24, 24) | conv2d_1 |
| 4 | conv2d_2(Conv2D) | (44, 24, 24) | max_pool2d_1 |
| 5 | max_pool2d_2(MaxPooling2D) | (44, 12, 12) | conv2d_2 |
| 6 | conv2d_3(Conv2D) | (44, 12, 12) | max_pool2d_2 |
| 7 | max_pool2d_3(MaxPooling2D) | (88, 6, 6) | conv2d_3 |
| 8 | dense_1(Dense) | (2048) | max_pool2d_3 |
| 9 | dropout_1(Dropout) | (2048) | dense_1 |
| 10 | dense_2(Dense) | (1024) | dropout_1 |
| 11 | dropout_2(Dropout) | (1024) | dense_2 |
| 12 | dense_3(Dense) | (8) | dropout_2 |

Table 3.15: The neural network described in the "Recognizing Facial Expressions Using a Shallow Convolutional Neural Network" paper.

Through the preliminary subdivision and the 9-fold, the following subdivision was obtained:

- 80% for the train dataset;
- 10% for the validation and the test datasets;

In the subsequent subsections, we will not talk again in-depth about the databases used, because we have already done so with the first network.

### FER2013 database

As first test, we chose to use the FER2013 database [18], which was also used in the paper to test the neural network. Using the configuration discussed above, we obtained the results that can be seen in table 3.16.

| **Fold #** | **Epochs of Training** | **Test Accuracy** | **Max Validation Accuracy** | **Min Validation Loss** | **Final Validation Accuracy** | **Final Validation Loss** |
|---|---|---|---|---|---|---|
| **0** | **26** | **0.54031** | **0.54345** | **1.32770** | **0.54011** | **2.66320** |
| 1 | 27 | 0.52971 | 0.53775 | 1.34140 | 0.52076 | 2.70367 |
| 2 | 28 | 0.52022 | 0.54639 | 1.33296 | 0.53580 | 2.67951 |
| 3 | 28 | 0.52831 | 0.53135 | 1.39962 | 0.52884 | 2.68709 |
| 4 | 28 | 0.52915 | 0.54556 | 1.33967 | 0.54277 | 2.59144 |
| 5 | 27 | 0.53110 | 0.53998 | 1.34322 | 0.53831 | 2.63350 |
| 6 | 28 | 0.52552 | 0.55280 | 1.33953 | 0.55280 | 2.58805 |
| 7 | 27 | 0.53026 | 0.53720 | 1.33277 | 0.52772 | 2.68649 |
| 8 | 27 | 0.53026 | 0.53720 | 1.34818 | 0.52382 | 2.64731 |

Table 3.16: Second network: results of the first training series with the FER2013 database, for every fold.

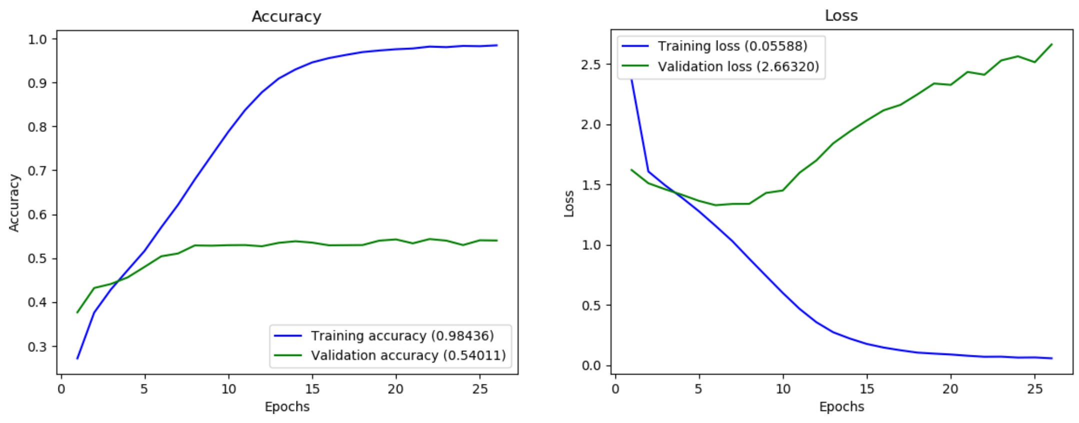


Figure 3.37: Second network, FER2013 database, first training series: loss and accuracy of the best network.

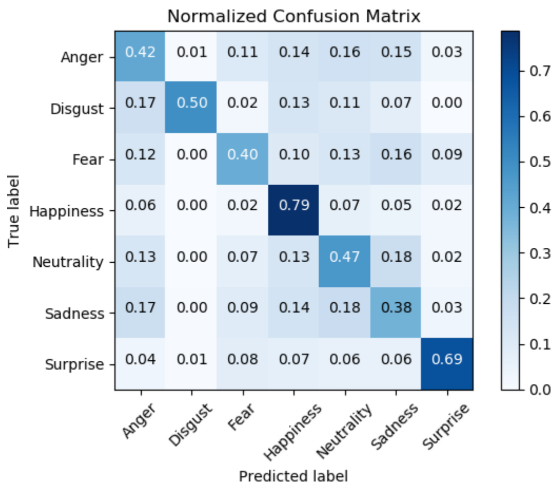


Figure 3.38: Second network, FER2013 database, first training series: normalized confusion matrix of the best network.

From the loss graph (figure 3.37) of the best network, obtained using the fold zero as validation dataset, we can see that there is a high overfitting at the end of the training: this is a problem that all the trained networks have, as we can see in table 3.16.

So, we tried to reduce this adding the scaling to a range normalization between 0 and 1, which, from now, we will call simply [0,1] normalization; we also added the data augmentation with the following parameters:

- a brightness range from 50% to 100%;
- horizontal flip enabled;
- rotation interval between +− 2.5 degrees;
- a shear range of +− 2.5%;
- a width and height shift range of +− 2.5%;
- a zoom transformation interval between +− 2.5%;

The results of the training are visible in table 3.17.

| Fold # | Epochs of Training | Test Accuracy | Max Validation Accuracy | Min Validation Loss | Final Validation Accuracy | Final Validation Loss |
|---|---|---|---|---|---|---|
| 0 | 30 | 0.61032 | 0.60696 | 1.14295 | 0.60696 | 1.53288 |
| 1 | 33 | 0.59944 | 0.61382 | 1.13490 | 0.60992 | 1.62046 |
| **2** | **32** | **0.61674** | **0.61159** | **1.13793** | **0.60518** | **1.63738** |
| 3 | 32 | 0.61311 | 0.60769 | 1.17814 | 0.60658 | 1.62133 |
| 4 | 30 | 0.59191 | 0.60602 | 1.14051 | 0.59710 | 1.57101 |
| 5 | 29 | 0.60139 | 0.60463 | 1.14682 | 0.58930 | 1.55101 |
| 6 | 32 | 0.59163 | 0.62301 | 1.12906 | 0.61744 | 1.55449 |
| 7 | 31 | 0.60809 | 0.60574 | 1.16365 | 0.60574 | 1.70865 |
| 8 | 30 | 0.59861 | 0.60769 | 1.14109 | 0.59877 | 1.60830 |

Table 3.17: Second network: results of the second training series with the FER2013 database, for every fold.

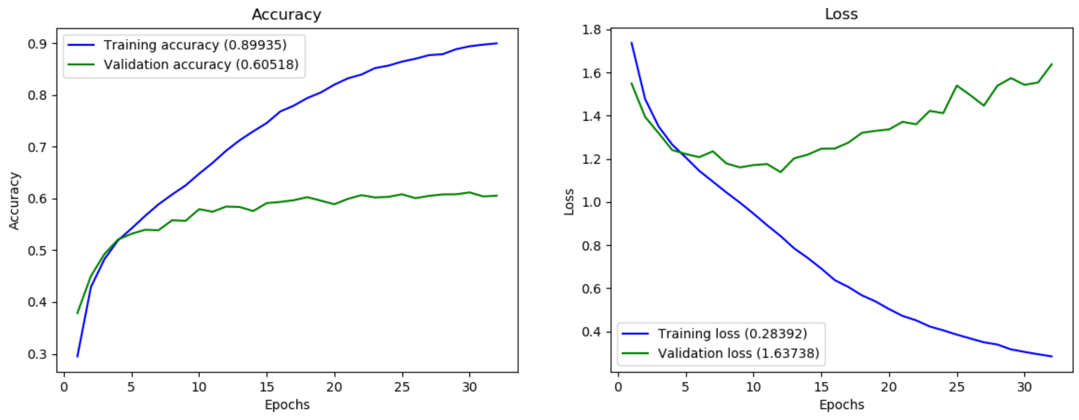
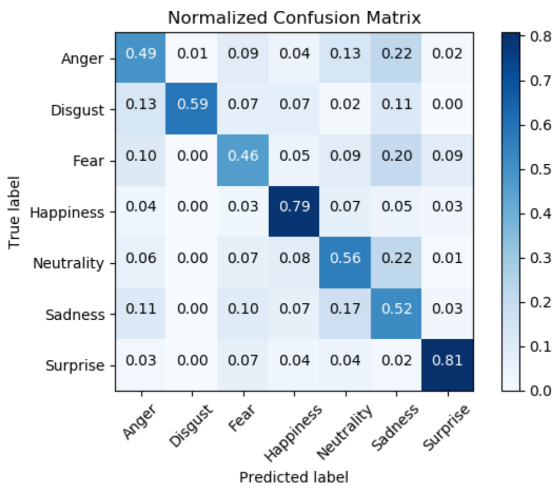


Figure 3.39: Second network, FER2013 database, second training series: loss and accuracy of the best network.



| True label \ Predicted label | Anger | Disgust | Fear | Happiness | Neutrality | Sadness | Surprise |
|---|---|---|---|---|---|---|---|
| Anger | 0.49 | 0.01 | 0.09 | 0.04 | 0.13 | 0.22 | 0.02 |
| Disgust | 0.13 | 0.59 | 0.07 | 0.07 | 0.02 | 0.11 | 0.00 |
| Fear | 0.10 | 0.00 | 0.46 | 0.05 | 0.09 | 0.20 | 0.09 |
| Happiness | 0.04 | 0.00 | 0.03 | 0.79 | 0.07 | 0.05 | 0.03 |
| Neutrality | 0.06 | 0.00 | 0.07 | 0.08 | 0.56 | 0.22 | 0.01 |
| Sadness | 0.11 | 0.00 | 0.10 | 0.07 | 0.17 | 0.52 | 0.03 |
| Surprise | 0.03 | 0.00 | 0.07 | 0.04 | 0.04 | 0.02 | 0.81 |

Figure 3.40: Second network, FER2013 database, second training series: normalized confusion matrix of the best network.

We can see that overfitting is still present, but reduced (figure 3.39); test accuracy also increased from 54.031% to 61.674%, with an improvement of about 7.5%. So, we did another test, replacing the [0,1] normalization with the z-score standardization: the results can be seen in table 3.18.

| **Fold #** | **Epochs of Training** | **Test Accuracy** | **Max Validation Accuracy** | **Min Validation Loss** | **Final Validation Accuracy** | **Final Validation Loss** |
|---|---|---|---|---|---|---|
| **0** | **28** | **0.62706** | **0.62507** | **1.11134** | **0.62507** | **1.49650** |
| 1 | 28 | 0.62064 | 0.62719 | 1.08190 | 0.62719 | 1.57904 |
| 2 | 27 | 0.60502 | 0.62497 | 1.10494 | 0.61633 | 1.51394 |
| 3 | 30 | 0.62287 | 0.61800 | 1.12345 | 0.61493 | 1.60090 |
| 4 | 28 | 0.61674 | 0.62162 | 1.12227 | 0.61577 | 1.46712 |
| 5 | 30 | 0.61618 | 0.61995 | 1.10043 | 0.61884 | 1.52056 |
| 6 | 29 | 0.62176 | 0.63360 | 1.09683 | 0.63360 | 1.46685 |
| 7 | 26 | 0.62232 | 0.61939 | 1.09767 | 0.60992 | 1.49174 |
| 8 | 28 | 0.61841 | 0.62190 | 1.11380 | 0.61326 | 1.61051 |

Table 3.18: Second network: results of the third training series with the FER2013 database, for every fold.

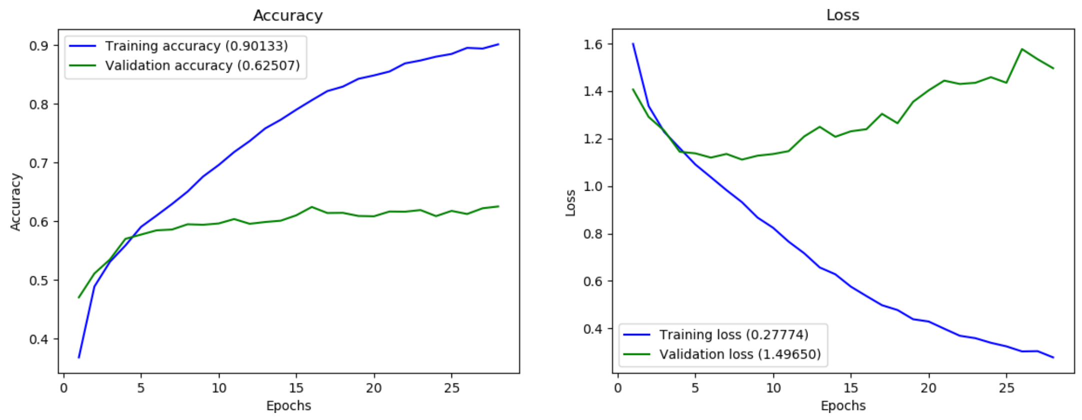


Figure 3.41: Second network, FER2013 database, third training series: loss and accuracy of the best network.

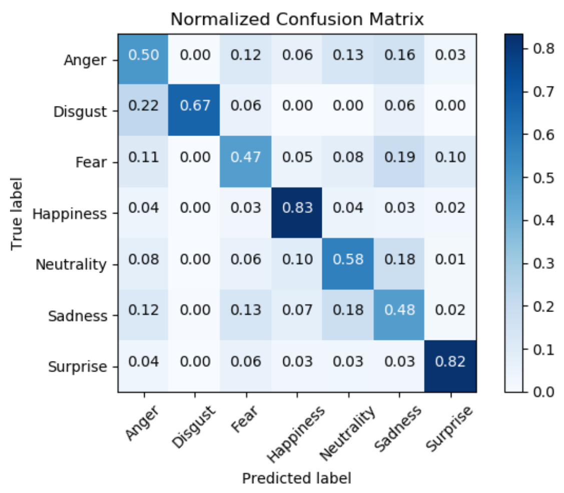


Figure 3.42: Second network, FER2013 database, third training series: normalized confusion matrix of the best network.

With the z-score standardization, the test accuracy has increased to 62.706%, about 1% more than before; the overfitting still persists, but on average it is a little smaller (figure 3.41). Using these same settings, which are the best we have achieved so far with the FER2013 database, and with this network, we have used the FER+ annotations to find out what the increase in test accuracy would have been.

| **Fold #** | **Epochs of Training** | **Test Accuracy** | **Max Validation Accuracy** | **Min Validation Loss** | **Final Validation Accuracy** | **Final Validation Loss** |
|---|---|---|---|---|---|---|
| 0 | 44 | 0.76051 | 0.74845 | 0.77456 | 0.74845 | 0.82788 |
| 1 | 45 | 0.73907 | 0.74451 | 0.77359 | 0.73408 | 0.88808 |
| 2 | 49 | 0.75205 | 0.75571 | 0.75651 | 0.74387 | 0.89038 |
| 3 | 50 | 0.75599 | 0.74866 | 0.76107 | 0.74134 | 0.87609 |
| **4** | **51** | **0.76671** | **0.75852** | **0.73327** | **0.75007** | **0.85974** |
| 5 | 43 | 0.74669 | 0.74556 | 0.79539 | 0.74162 | 0.87022 |
| 6 | 52 | 0.75205 | 0.75824 | 0.74593 | 0.75092 | 0.86451 |
| 7 | 50 | 0.76164 | 0.74979 | 0.77193 | 0.74387 | 0.90470 |
| 8 | 46 | 0.74302 | 0.75035 | 0.78157 | 0.73711 | 0.87906 |

Table 3.19: Second network: results of the forth training series with the FER2013 database with the FER+ annotations, for every fold.

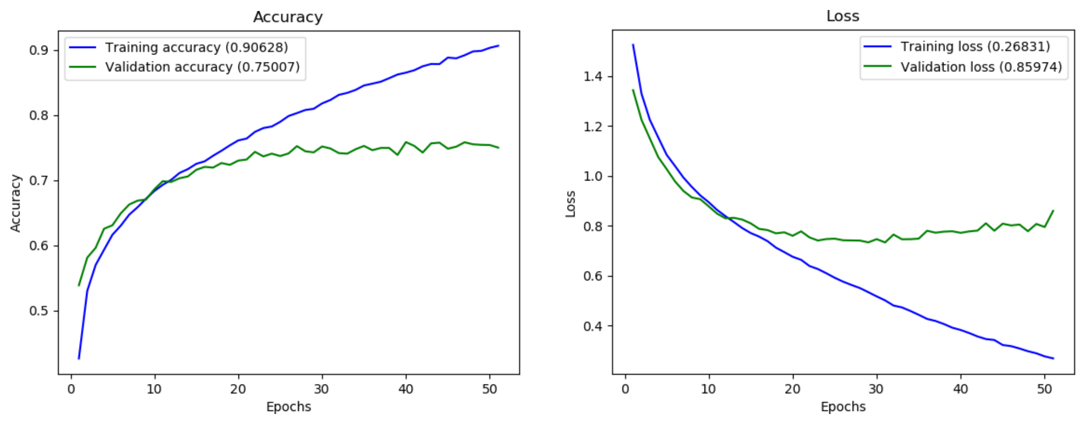


Figure 3.43: Second network, FER2013 database with the FER+ annotations, forth training series: loss and accuracy of the best network.

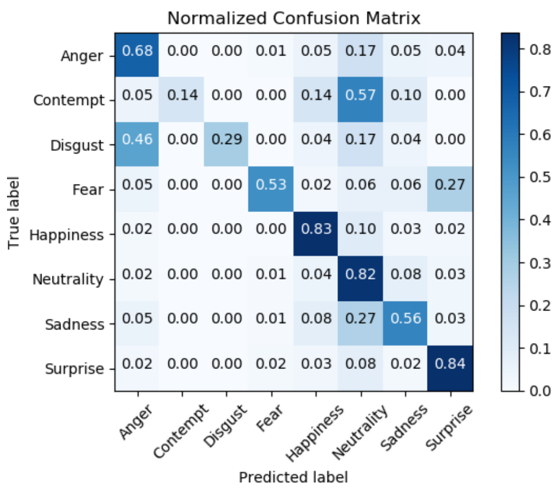


Figure 3.44: Second network, FER2013 database with the FER+ annotations, forth training series: normalized confusion matrix of the best network.

The best network achieved an accuracy of 76.671% (figure 3.19), that is a good result, however far from the state of the art and moreover about 2% lower than the best result obtained with the other neural network. By carrying out more tests, trying different values for hyperparameters and using other images processing techniques, it is certainly possible to increase this value. Like the previous network, from the confusion matrix (figure 3.44) it can be seen that the performance with the cases of contempt and disgust are very bad, probably due to the low number of images available for these facial expressions.

### Database Ensemble of all the facial expressions databases supported by FEDC

We decided to carry out other tests, with this network, using the Database Ensemble composed of all the supported databases of FEDC, which, with its approximately 44k photos, reached just over 80% with the previous network, around 2% more than FER2013 with the FER+ annotations. Therefore, we wanted to know the performance that this network could achieve with the same database, so we started a first series of training, using the default settings.

| Fold # | Epochs of Training | Test Accuracy | Max Validation Accuracy | Min Validation Loss | Final Validation Accuracy | Final Validation Loss |
|---|---|---|---|---|---|---|
| 0 | 30 | 0.70473 | 0.71422 | 0.94393 | 0.70763 | 1.56618 |
| 1 | 33 | 0.70724 | 0.70309 | 0.96692 | 0.70195 | 1.68587 |
| 2 | 31 | 0.70792 | 0.70150 | 0.95007 | 0.70082 | 1.59655 |
| 3 | 31 | 0.70610 | 0.71876 | 0.94849 | 0.70332 | 1.50601 |
| 4 | 33 | 0.69742 | 0.72490 | 0.92732 | 0.70718 | 1.47552 |
| **5** | **31** | **0.71386** | **0.70825** | **0.93753** | **0.70825** | **1.47951** |
| 6 | 30 | 0.70815 | 0.71643 | 0.94069 | 0.70189 | 1.60602 |
| 7 | 31 | 0.70975 | 0.71188 | 0.95577 | 0.71029 | 1.55642 |
| 8 | 32 | 0.71203 | 0.70166 | 0.98792 | 0.70075 | 1.63287 |

Table 3.20: Second network: results of the first training series with the Database Ensemble of all the facial expressions databases, for every fold.

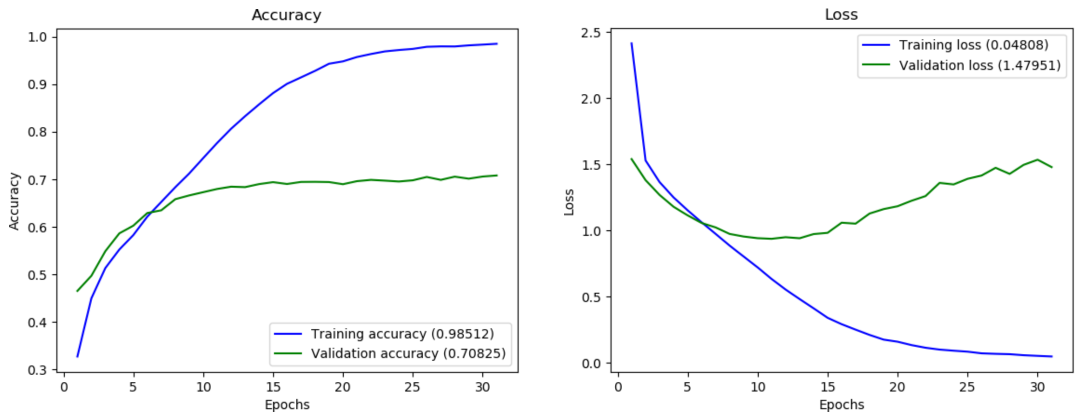


Figure 3.45: Second network, Database Ensemble of all the facial expressions databases, first training series: loss and accuracy of the best network.

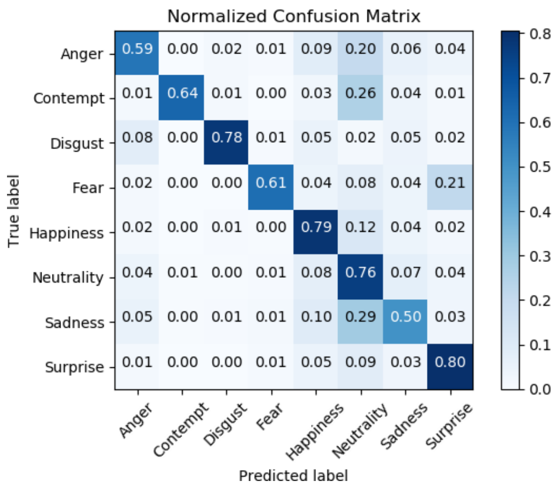


Figure 3.46: Second network, Database Ensemble of all the facial expressions databases, first training series: normalized confusion matrix of the best network.

The best network obtained a test accuracy of 71.386% (table 3.20). A consistent overfitting is present towards the end of the training (figure 3.45), so, methods for reducing it are needed: adding the [0,1] normalization and the data augmentation, with the same parameters that we listed before, we obtained the result in table 3.21.

| Fold # | Epochs of Training | Test Accuracy | Max Validation Accuracy | Min Validation Loss | Final Validation Accuracy | Final Validation Loss |
|---|---|---|---|---|---|---|
| 0 | 67 | 0.75405 | 0.76420 | 0.72287 | 0.75988 | 0.76959 |
| 1 | 66 | 0.75383 | 0.75852 | 0.74369 | 0.75125 | 0.77098 |
| 2 | 67 | 0.75451 | 0.75670 | 0.74199 | 0.75670 | 0.78084 |
| 3 | 62 | 0.75200 | 0.76193 | 0.73144 | 0.75261 | 0.78043 |
| 4 | 60 | 0.75634 | 0.75375 | 0.76255 | 0.75193 | 0.78797 |
| 5 | 78 | 0.76867 | 0.77142 | 0.70332 | 0.76869 | 0.79194 |
| 6 | 63 | 0.75360 | 0.75869 | 0.74453 | 0.75687 | 0.77810 |
| 7 | 72 | 0.75679 | 0.76142 | 0.74951 | 0.75846 | 0.81412 |
| **8** | **75** | **0.76913** | **0.76846** | **0.72408** | **0.76460** | **0.80526** |

Table 3.21: Second network: results of the second training series with the Database Ensemble of all the facial expressions databases, for every fold.

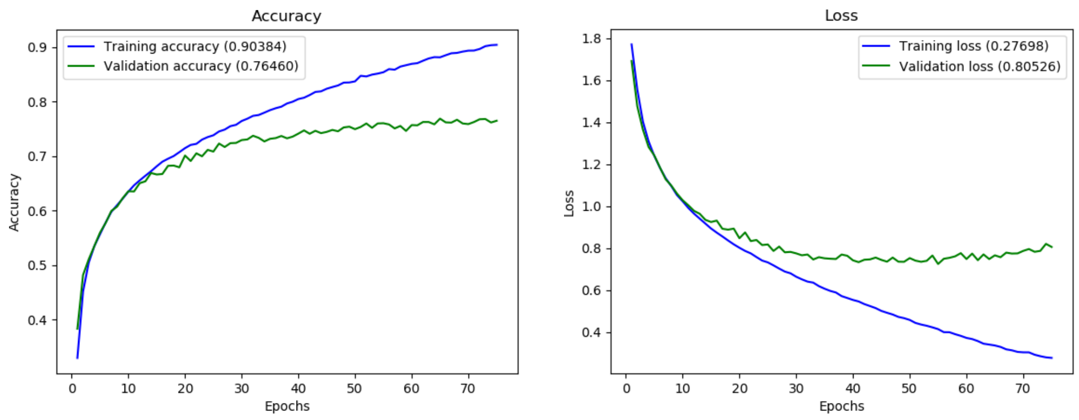


Figure 3.47: Second network, Database Ensemble of all the facial expressions databases, second training series: loss and accuracy of the best network.

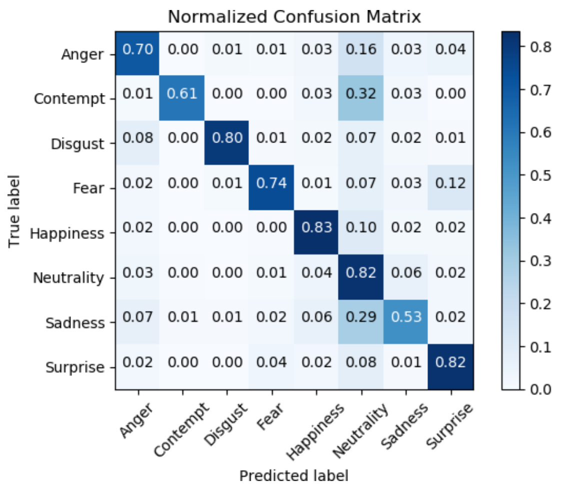


Figure 3.48: Second network, Database Ensemble of all the facial expressions databases, second training series: normalized confusion matrix of the best network.

The best network, obtained using the eighth fold as validation dataset, has achieved a test accuracy of 76.913%, having much less overfitting than before (figure 3.47). As last test with this network, we have replaced the [0,1] normalization with the z-score standardization.

| Fold # | Epochs of Training | Test Accuracy | Max Validation Accuracy | Min Validation Loss | Final Validation Accuracy | Final Validation Loss |
|---|---|---|---|---|---|---|
| 0 | 55 | 0.77575 | 0.77806 | 0.70743 | 0.77442 | 0.79705 |
| 1 | 48 | 0.76136 | 0.76920 | 0.73583 | 0.75943 | 0.82329 |
| 2 | 54 | 0.76205 | 0.76102 | 0.75215 | 0.75920 | 0.89250 |
| 3 | 50 | 0.76707 | 0.77260 | 0.72821 | 0.76670 | 0.80878 |
| 4 | 47 | 0.77072 | 0.76397 | 0.74732 | 0.76124 | 0.83240 |
| 5 | 49 | 0.76935 | 0.76210 | 0.70761 | 0.76142 | 0.80077 |
| 6 | 44 | 0.77483 | 0.76528 | 0.73239 | 0.76051 | 0.79613 |
| 7 | 46 | 0.77392 | 0.76369 | 0.74481 | 0.75801 | 0.80058 |
| **8** | **58** | **0.78465** | **0.77641** | **0.71401** | **0.77414** | **0.81141** |

Table 3.22: Second network: results of the third training series with the posed and spontaneous facial expressions Database Ensemble, for every fold.

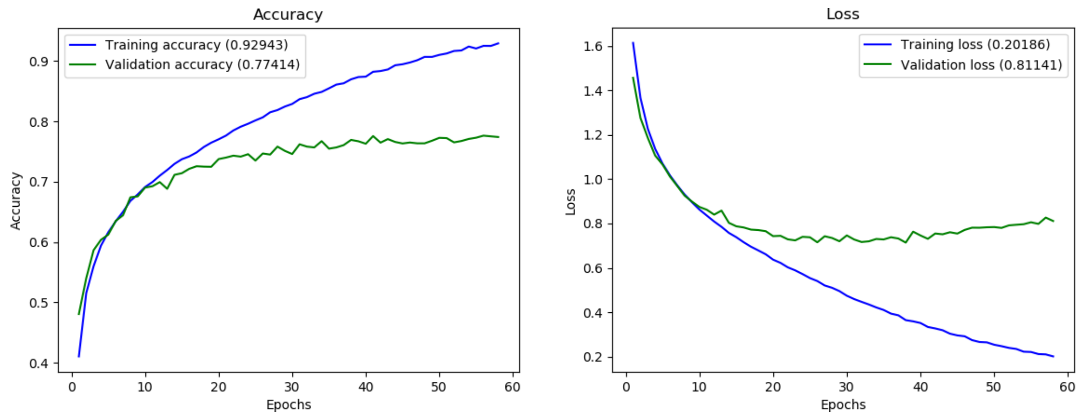
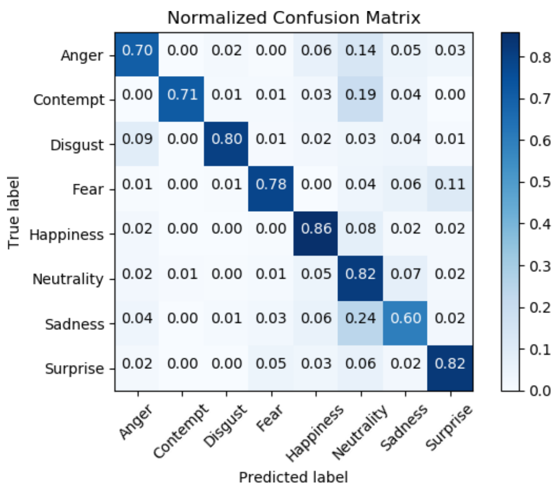


Figure 3.49: Second network, Database Ensemble of all the facial expressions databases, third training series: loss and accuracy of the best network.



|  | Anger | Contempt | Disgust | Fear | Happiness | Neutrality | Sadness | Surprise |
|---|---|---|---|---|---|---|---|---|
| Anger | 0.70 | 0.00 | 0.02 | 0.00 | 0.06 | 0.14 | 0.05 | 0.03 |
| Contempt | 0.00 | 0.71 | 0.01 | 0.01 | 0.03 | 0.19 | 0.04 | 0.00 |
| Disgust | 0.09 | 0.00 | 0.80 | 0.01 | 0.02 | 0.03 | 0.04 | 0.01 |
| Fear | 0.01 | 0.00 | 0.01 | 0.78 | 0.00 | 0.04 | 0.06 | 0.11 |
| Happiness | 0.02 | 0.00 | 0.00 | 0.00 | 0.86 | 0.08 | 0.02 | 0.02 |
| Neutrality | 0.02 | 0.01 | 0.00 | 0.01 | 0.05 | 0.82 | 0.07 | 0.02 |
| Sadness | 0.04 | 0.00 | 0.01 | 0.03 | 0.06 | 0.24 | 0.60 | 0.02 |
| Surprise | 0.02 | 0.00 | 0.00 | 0.05 | 0.03 | 0.06 | 0.02 | 0.82 |

Figure 3.50: Second network, Database Ensemble of all the facial expressions databases, third training series: normalized confusion matrix of the best network.

With this last training series, the accuracy has increased to 78.465% (table 3.22), about 1.5% better than the last result, but 2% lower than the top result of the other network.

### Database Ensemble of all the posed facial expressions databases supported by FEDC

As the last series of tests, we decided to use, as database, the Database Ensemble of all the posed facial expressions databases supported by FEDC. The default images size of the database is 120x120 pixels, but, using this dimension, the number of parameters required by the network exceeded 42 million: a number out of the question for our task. So, we used an images size of 48x48 pixels. The first training series were made with the same default settings that we have used so far.

| **Fold #** | **Epochs of Training** | **Test Accuracy** | **Max Validation Accuracy** | **Min Validation Loss** | **Final Validation Accuracy** | **Final Validation Loss** |
|---|---|---|---|---|---|---|
| 0 | 70 | 0.92168 | 0.90971 | 0.35392 | 0.89603 | 0.39677 |
| 1 | 65 | 0.89930 | 0.89603 | 0.38485 | 0.88919 | 0.44421 |
| 2 | 50 | 0.89930 | 0.90287 | 0.35048 | 0.88782 | 0.40239 |
| 3 | 57 | 0.91608 | 0.89466 | 0.39016 | 0.89466 | 0.44771 |
| **4** | **67** | **0.92587** | **0.89740** | **0.33108** | **0.88509** | **0.40613** |
| 5 | 65 | 0.89930 | 0.88919 | 0.37963 | 0.87825 | 0.47014 |
| 6 | 70 | 0.91888 | 0.88235 | 0.44195 | 0.87688 | 0.47819 |
| 7 | 50 | 0.91748 | 0.88509 | 0.44123 | 0.87688 | 0.49250 |
| 8 | 62 | 0.91888 | 0.88630 | 0.41992 | 0.88356 | 0.46755 |

Table 3.23: Second network: results of the first training series with the Database Ensemble of posed facial expressions, for every fold.

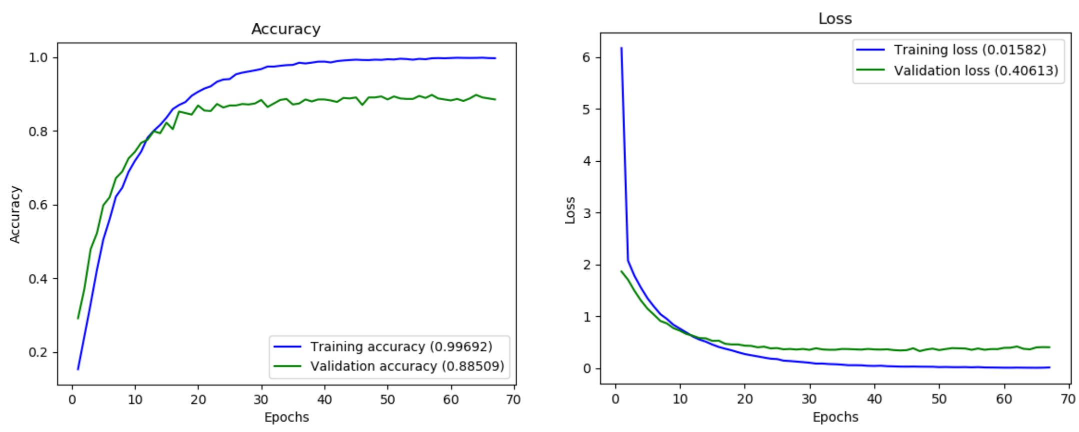


Figure 3.51: Second network, Database Ensemble of posed facial expressions, first training series: loss and accuracy of the best network.

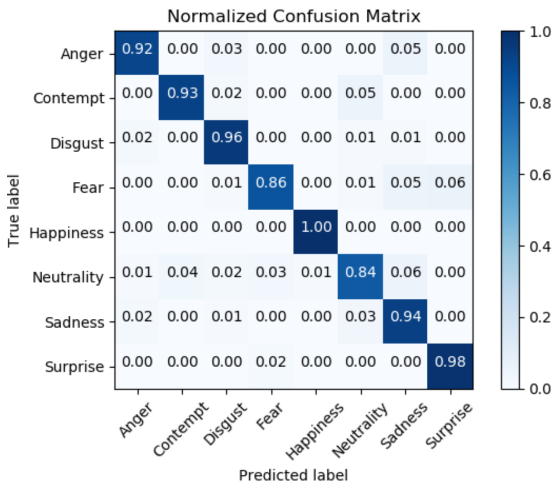


Figure 3.52: Second network, Database Ensemble of posed facial expressions, first training series: normalized confusion matrix of the best network.

The best network, obtained using the fourth fold as validation dataset (table 3.23), has reached an accuracy of 92.587%. In general, for all folds, the results are quite good, and we can see from the graphs of the best network that both the accuracy and the validation loss tend to saturate (figure 3.51). We then did another series of tests, adding the data augmentation and the [0,1] normalization.

| **Fold #** | **Epochs of Training** | **Test Accuracy** | **Max Validation Accuracy** | **Min Validation Loss** | **Final Validation Accuracy** | **Final Validation Loss** |
|---|---|---|---|---|---|---|
| 0 | 160 | 0.92727 | 0.92202 | 0.26021 | 0.91655 | 0.27167 |
| 1 | 186 | 0.93427 | 0.92202 | 0.26615 | 0.90834 | 0.28135 |
| 2 | 160 | 0.92867 | 0.91382 | 0.26997 | 0.90834 | 0.28574 |
| 3 | 146 | 0.91888 | 0.91108 | 0.27067 | 0.90150 | 0.28854 |
| 4 | 184 | 0.93566 | 0.90834 | 0.27611 | 0.89330 | 0.32516 |
| **5** | **179** | **0.94825** | **0.92339** | **0.23256** | **0.91245** | **0.25431** |
| 6 | 210 | 0.93287 | 0.90834 | 0.29135 | 0.90834 | 0.33928 |
| 7 | 130 | 0.90909 | 0.89056 | 0.36083 | 0.88782 | 0.36966 |
| 8 | 159 | 0.92028 | 0.90822 | 0.31355 | 0.88630 | 0.35776 |

Table 3.24: Second network: results of the second training series with the Database Ensemble of posed facial expressions, for every fold.

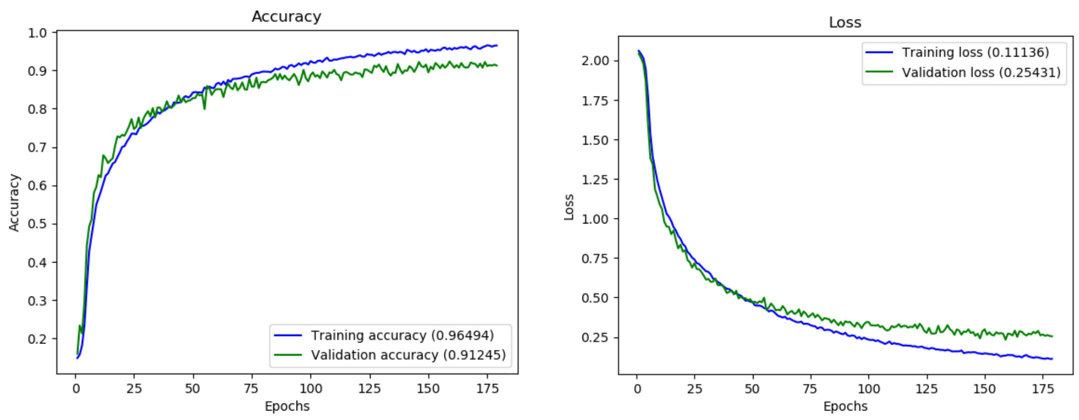


Figure 3.53: Second network, Database Ensemble of posed facial expressions, second training series: loss and accuracy of the best network.

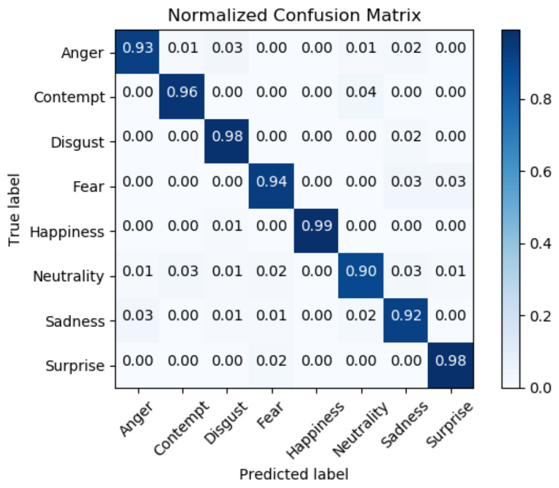


Figure 3.54: Second network, Database Ensemble of posed facial expressions, second training series: normalized confusion matrix of the best network.

We can see, from the table 3.53, that many epochs were necessary to train the network with these parameters. Test accuracy has increased slightly, rising to 94.825% for the best network. As the last series of tests, we have replaced the [0,1] normalization with the z-score standardization.

| Fold # | Epochs of Training | Test Accuracy | Max Validation Accuracy | Min Validation Loss | Final Validation Accuracy | Final Validation Loss |
|---|---|---|---|---|---|---|
| 0 | 140 | 0.94965 | 0.94118 | 0.19601 | 0.92613 | 0.24151 |
| 1 | 119 | 0.95524 | 0.93434 | 0.23120 | 0.92613 | 0.25884 |
| 2 | 118 | 0.95524 | 0.92750 | 0.25967 | 0.91792 | 0.28674 |
| 3 | 208 | 0.95944 | 0.95212 | 0.18458 | 0.94665 | 0.20879 |
| **4** | **121** | **0.96783** | **0.92750** | **0.24785** | **0.92066** | **0.29790** |
| 5 | 150 | 0.95105 | 0.93981 | 0.19217 | 0.93981 | 0.19316 |
| 6 | 125 | 0.94685 | 0.92202 | 0.27961 | 0.92202 | 0.28979 |
| 7 | 116 | 0.95105 | 0.92476 | 0.28561 | 0.90561 | 0.32591 |
| 8 | 145 | 0.95524 | 0.94247 | 0.22367 | 0.92740 | 0.26000 |

Table 3.25: Second network: results of the third training series with the Database Ensemble of posed facial expressions, for every fold.

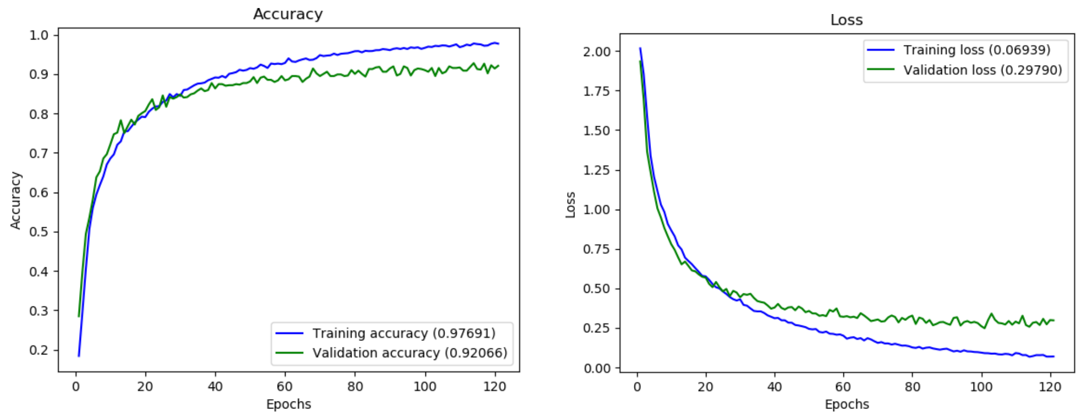
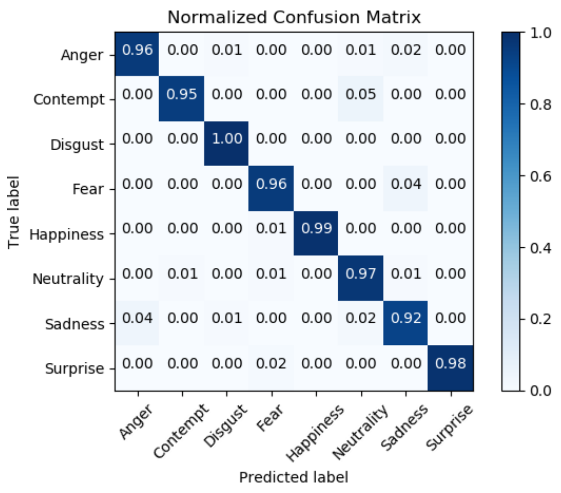


Figure 3.55: Second network, Database Ensemble of posed facial expressions, third training series: loss and accuracy of the best network.



| True label | Anger | Contempt | Disgust | Fear | Happiness | Neutrality | Sadness | Surprise |
|---|---|---|---|---|---|---|---|---|
| Anger | 0.96 | 0.00 | 0.01 | 0.00 | 0.00 | 0.01 | 0.02 | 0.00 |
| Contempt | 0.00 | 0.95 | 0.00 | 0.00 | 0.00 | 0.05 | 0.00 | 0.00 |
| Disgust | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Fear | 0.00 | 0.00 | 0.00 | 0.96 | 0.00 | 0.00 | 0.04 | 0.00 |
| Happiness | 0.00 | 0.00 | 0.00 | 0.01 | 0.99 | 0.00 | 0.00 | 0.00 |
| Neutrality | 0.00 | 0.01 | 0.00 | 0.01 | 0.00 | 0.97 | 0.01 | 0.00 |
| Sadness | 0.04 | 0.00 | 0.01 | 0.00 | 0.00 | 0.02 | 0.92 | 0.00 |
| Surprise | 0.00 | 0.00 | 0.00 | 0.02 | 0.00 | 0.00 | 0.00 | 0.98 |

Figure 3.56: Second network, Database Ensemble of posed facial expressions, third training series: normalized confusion matrix of the best network.

With this last series of tests, the best network reached a test accuracy of 96.783% (table 3.25), about 2% more than the previous result, but lower than the 97.343% achieved by the first network.

### 3.7.3 Comments on the results obtained

We analyzed the networks present in two different papers:

- **Physiological Inspired Deep Neural Networks for Emotion Recognition** [17];
- **Recognizing Facial Expressions Using a Shallow Convolutional Neural Network** [27].

In general, we can see that these two networks, which we have chosen and tested, have very similar results, but, as we have seen in the last section of each test series, the first has almost always had slightly better results. If we were to choose one, we would certainly choose the first, because, although it is more complex and requires more time for training, it works better, and, for the same images size, it requires far fewer parameters: with 48x48 pixels images, it requires little more of 2 million, against the approximately 9 million of the other network. This factor is very important to us, because in the future it will be necessary to implement this neural network in a single board computer.

Page intentionally left blank.

# Chapter 4

# Road tests

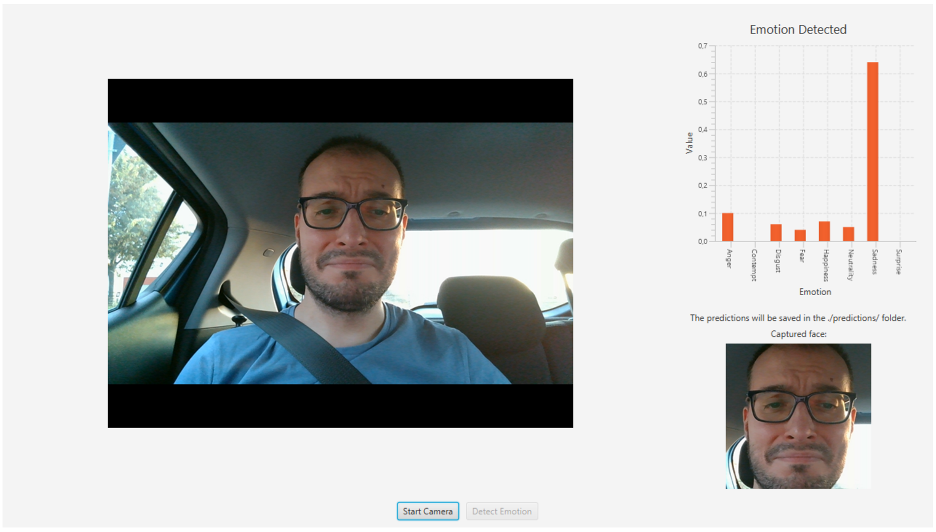
*After training some neural networks, we have reached the point of wanting to do some road tests, so as to get evidence on how they work in practical cases. Since we did not have the opportunity to access to an autonomous driving vehicle, we decided to do a feasibility study, using a normal car and capturing images in the vehicle with a webcam of a laptop or a tablet, in different weather conditions and lights. In order to make this type of tests, it was necessary to develop a software able to execute the neural network and to display the results on-the-fly.*

## 4.1 Emotion Detector

For the computer vision exam, besides the Data Preparation for CK+ and Data Preparation for FER2013 program, we realized, resorting to Eclipse, with the addition of **Java**, **OpenCV** [9], **DeepLearning4J** (acronym **DL4J**) [52] and **Apache Maven** another program, called **Emotion Detector**. This program, at the user's request, is able to acquire frames from a webcam of a laptop or desktop computer, and, again upon request, can use a neural network to make a prediction about the user's current facial expression. Once the analysis is finished, the results are arranged on a histogram, so that the user can view and interpret them in a simple way. Emotion Detection was a natural choice for the task we had to perform, so we have therefore decided to improve its interface by adding:

- a box, in which to show the image as it is seen by the neural network, and on which it will make the prediction; so, this box will only contain the face of the user, cropped with the **Haar Feature-based Cascade Classifiers**;

- a function to automatically take a photo of the program interface immediately after a prediction, in order to collect and document a series of actual results.

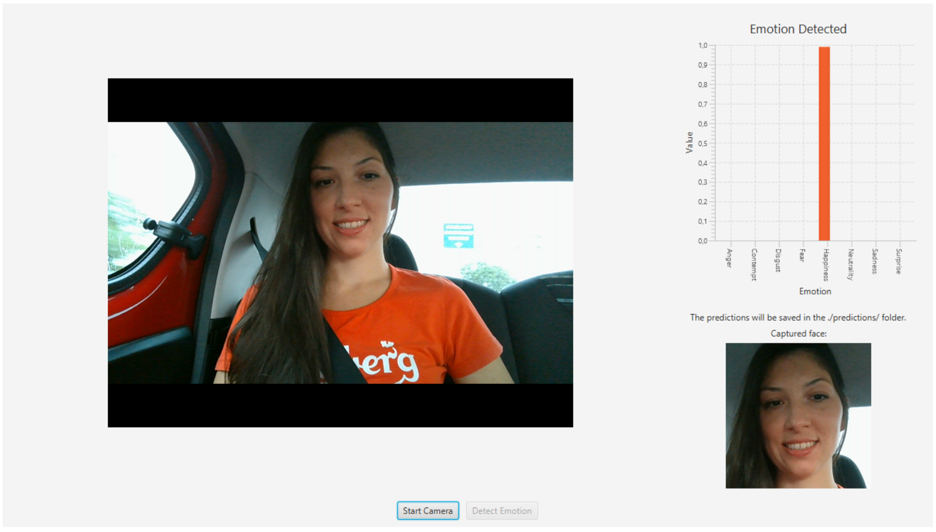


The predictions will be saved in the ./predictions/ folder.

Figure 4.1: One of the photos taken during the road tests.

## 4.2 Tests

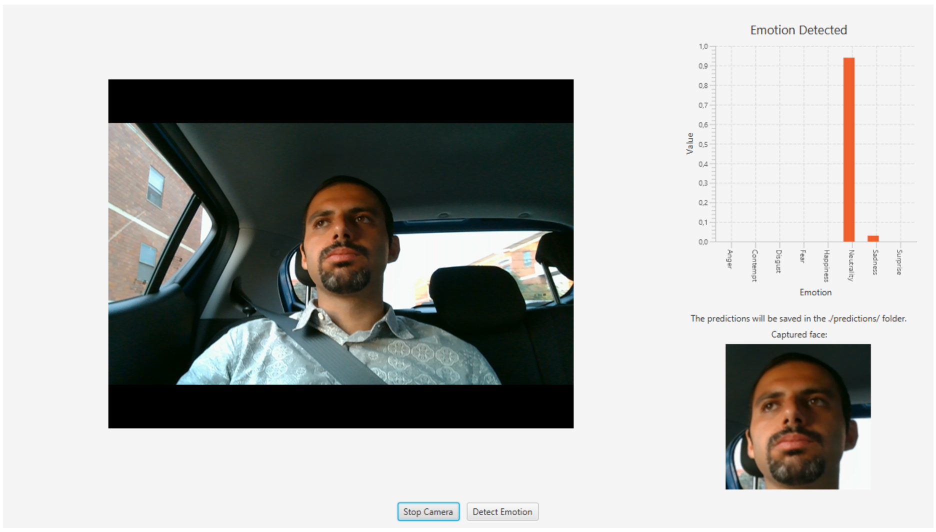
After making the modifications described above to Emotion Detection, and obtaining some neural networks with a good test accuracy, we made multiple test sessions in practical conditions, with different actors and cars: we sat the actor/actress on the back seats of the car, with a laptop or a tablet on his/her legs, raised with a special pedestal in order to be at the right face height.



The predictions will be saved in the ./predictions/ folder.

Figure 4.2: One of the photos taken during the road tests.

The first test session was made with the network described in the **Physiological Inspired Deep Neural Networks for Emotion Recognition** paper, trained with the **FER2013 database** [18] with the **FER+ annotations** [8]: the problem is that this is not a well-balanced database, and, as we have seen before from the confusion matrix in figure 3.21, it has a very low performance with some expressions: so, it was difficult to achieve a part of them, like contempt or disgust, but it had a strong recognition of other expressions, like happiness or neutral.

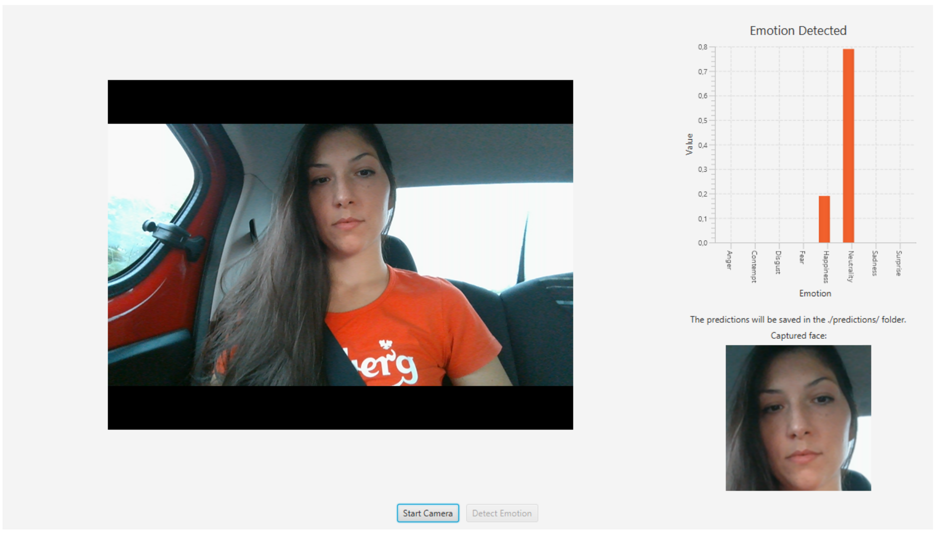


The predictions will be saved in the ./predictions/ folder.

Figure 4.3: One of the photos taken during the road tests.

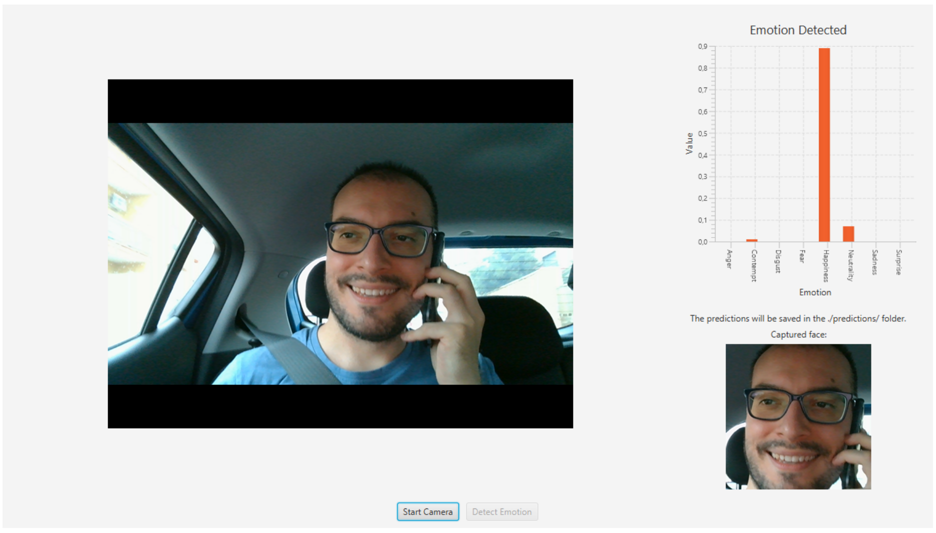
We must remember that, for our project, we do not need all these facial expressions, but we are only interested in **fear**, **happiness**, **neutrality**, **sadness** and **surprise**; anger, contempt and disgust will be treated by the system as "**do not care**", because they are useless for controlling the speed of the vehicle. Maybe, they can be used for other purposes, for example for ambient intelligence, but these applications are beyond the purpose of this thesis.

We carried out other tests using the same neural network, but trained with the **Database Ensemble of all the facial expressions databases supported by FEDC**, obtaining better results. Due to the high number of neutral and happiness photos, even here it was easy to get these two states, but a greater number of photos of contempt, disgust, fear and sadness states made the recognition of these facial expressions a little more accurate: this can be seen also from the figure 3.29, depicting the confusion matrix of the neural network.



The predictions will be saved in the ./predictions/ folder.

Figure 4.4: One of the photos taken during the road tests.



The predictions will be saved in the ./predictions/ folder.

Figure 4.5: One of the photos taken during the road tests.

# Chapter 5

# Conclusions

## 5.1 Results

A feasibility study was conducted in this thesis. In order to obtain useful results, we have tried to detail and innovate all the sections encountered by our work. In particular, we have carried out the following works:

- we developed a novel software, called **Facial Expressions Databases Classifier** (**FEDC**) [48], that has a GUI that makes it simple to use, and that is capable to classify, for now, the images coming from seven facial expressions databases, applying optional transformations, chosen by the user, to each one of them. This reduces the work necessary for the pre-classification of the photos, the code necessary to train the neural networks and eases the creation of what we called **Database Ensemble**, that is a wider database obtained by merging the databases together. The code of FEDC is freely downloadable from GitHub, where it was distributed under the MIT license, which was specifically chosen to maximize the freedom given to those who wants to modify it. One of the strengths of the code is also the ease of modification, so that new databases and functions can be added;

- we tested some state-of-the-art neural networks, displaying the results obtained with the chosen hyperparameters and with commonly used operations to improve the convergence and performance of neural networks;

- we developed a program to be used for the on-the-field tests, called **Emotion Detector**, so as to assess the performances of some of the best neural networks we trained in cases similar to the real ones, with all the limits dictated by the absence of a real autonomous driving car.

After the tests, we think that the results we obtained are really promising: this approach, together with others that have been developed or are in-the-work, could really facilitate the diffusion of the autonomous driving cars as it was for the automation of trains and other types of rail transports, which are commonly used in everyday life.

## 5.2 Future work

For many years, national laws were strong constraints regard autonomous driving car tests, but, from a few years, things are changing. In this year, 2019, also in Italy there were the first authorizations to road tests regarding autonomous driving car; this is encouraging news, which gives us the right spirit for continuing our work in the near future.

So, our next step will be a transposition of this system into a single-board computer, like a Raspberry Pi 3, in order to test the performance of the neural network and evaluate the impact of existing optimization techniques, such as pruning and compression of neural networks, on a device with less computing power than a normal computer. We also want to work to improve FEDC [48], adding support for new databases and features.

The trainings were made with a relatively low number of images, a common problem for the neural networks: it could be useful, for future tests in controlled environments, to search for the participation of the testers: they can help to improve the accuracy of the recognition sharing the photos made by the cars, so as to be labeled by professionals and used to retrain the network, partially with fine-tuning or transfer learning, or in full: afterward, the new network can be redistributed as an update.

Page intentionally left blank.

Page intentionally left blank.

# References

## References from books

[1] Daniel Cordaro, Rui Sun, Dacher Keltner, Shanmukh Kamble, Niranjan Huddar, and Galen McNeil. Universals and cultural variations in 22 emotional expressions across five cultures. *Emotion*, 18, 06 2017.

[2] P Ekman and W Friesen. *Facial Action Coding System (FACS): A technique for the measurement of facial action.* 01 1978.

[3] Paul Ekman and Wallace V. Friesen. Constants across cultures in the face and emotion. *Journal of personality and social psychology*, 17 2:124-9, 1971.

[4] Ying Li, Takeo Kanade, Jeffrey Cohn, Stan Li, and Anil Jain. *Facial Expression Analysis*, pages 247–275. 12 2005.

[5] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[6] Kai Ming Ting. *Confusion Matrix*, pages 209-209. Springer US, Boston, MA, 2010.

## References from articles

[7] N. Aifanti, C. Papachristou, and A. Delopoulos. The mug facial expression database. In *11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10*, pages 1-4, 4 2010.

[8] Emad Barsoum, Cha Zhang, Cristian Ferrer, and Zhengyou Zhang. Training deep networks for facial expression recognition with crowd-sourced label distribution. pages 279-283, 08 2016.

[9] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[10] Kumar Chellapilla, Sidd Puri, and Patrice Simard. High performance convolutional neural networks for document processing. 10 2006.

[11] Dan C. Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *IJCAI*, 2011.

[12] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon. Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 2106-2112, 11 2011.

[13] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon. Collecting large, richly annotated facial-expression databases from movies. *IEEE MultiMedia*, 19(3):34-41, 7 2012.

[14] Abhinav Dhall, Roland Goecke, Jyoti Joshi, Karan Sikka, and Tom Gedeon. Emotion recognition in the wild challenge 2014: Baseline, data and protocol. 11 2014.

[15] Natalie Ebner, Michaela Riediger, and Ulman Lindenberger. Faces—a database of facial expressions in young, middle-aged, and older women and men: Development and validation. *Behavior research methods*, 42:351-62, 02 2010.

[16] P. Ekman. Basic emotions. In *Handbook of Cognition and Emotion*, pages 45-60, 1999.

[17] P. M. Ferreira, F. Marques, J. S. Cardoso, and A. Rebelo. Physiological inspired deep neural networks for emotion recognition. *IEEE Access*, 6:53930-53943, 2018.

[18] Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. Challenges in representation learning: A report on three machine learning contests, 2013.

[19] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9(3):90-95, 5 2007.

[20] T. Kanade, J. F. Cohn, and Yingli Tian. Comprehensive database for facial expression analysis. In *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pages 46-53, 3 2000.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[22] Oliver Langner, Ron Dotsch, Gijsbert Bijlstra, Daniel Wigboldus, Skyler Hawk, and Ad Knippenberg. Presentation and validation of the radboud face database. *Cognition & Emotion - COGNITION EMOTION*, 24:1377-1388, 12 2010.

[23] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[24] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. 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, 6 2010.

[25] 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, 4 1998.

[26] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 - 56, 2010.

[27] S. Miao, H. Xu, Z. Han, and Y. Zhu. Recognizing facial expressions using a shallow convolutional neural network. *IEEE Access*, 7:78000-78011, 2019.

[28] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825-2830, 2011.

[29] S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13(2):22-30, 3 2011.

## References from sites

All sites were last visited on October 5, 2019.

[30] Classification: accuracy. `https://developers.google.com/machine-learning/crash-course/classification/accuracy`.

[31] Descending into ML: training and loss. https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss.

[32] Keras FAQ: Frequently Asked Keras Questions. https://keras.io/getting-started/faq/.

[33] Loss functions: categorical crossentropy. https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy.

[34] Normalization. https://developers.google.com/machine-learning/data-prep/transform/normalization.

[35] NVIDIA cuDNN. https://developer.nvidia.com/cudnn.

[36] Training and test sets: splitting data. https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data.

[37] Usage of callbacks. https://keras.io/callbacks/.

[38] Validation set: another partition. https://developers.google.com/machine-learning/crash-course/validation/another-partition.

[39] 2018 Kaggle ML & DS survey. https://www.kaggle.com/kaggle/kaggle-survey-2018, 2018.

[40] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from https://www.tensorflow.org/.

[41] Anas Al-Masri. What are overfitting and underfitting in machine learning? https://towardsdatascience.com/what-are-overfitting-and-underfitting-in-machine-learning-a96b30864690, 06 2019.

[42] Jason Brownlee. How to configure image data augmentation in Keras. https://machinelearningmastery.com/reproducible-results-neural-networks-keras/, 06 2017.

[43] Jason Brownlee. How to configure image data augmentation in Keras. https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/, 04 2019.

[44] François Chollet et al. Keras. https://keras.io, 2015.

[45] History.com Editors. Automobile history. https://www.history.com/topics/inventions/automobiles, 04 2010.

[46] Josh.ai. Everything you need to know about artificial neural networks. https://medium.com/technology-invention-and-more/everything-you-need-to-know-about-artificial-neural-networks-57fac18245a1, 12 2015.

[47] Renu Khandelwal. K-fold and other cross-validation techniques. https://medium.com/datadriveninvestor/k-fold-and-other-cross-validation-techniques-6c03a2563f1e, 11 2018.

[48] A. C. Marceddu. Facial Expressions Databases Classifier (FEDC). https://github.com/AntonioMarceddu/Facial_Expressions_Databases_Classifier, 2019.

[49] Heads or Tails. What we do in the kernels - a Kaggle survey story. https://www.kaggle.com/headsortails/what-we-do-in-the-kernels-a-kaggle-survey-story, 12 2018.

[50] Seema Singh. Understanding the bias-variance tradeoff. https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229, 05 2018.

[51] Kate Strachnyi. Brief history of neural networks. https://medium.com/analytics-vidhya/brief-history-of-neural-networks-44c2bf72eec, 01 2019.

[52] Eclipse Deeplearning4j Development Team. Deeplearning4j: Open-source distributed deep learning for the JVM, Apache Software Foundation License 2.0. http://deeplearning4j.org/, 2019.

[53] Techopedia.com. Autonomous car. https://www.techopedia.com/definition/30056/autonomous-car.

[54] Treccani.it. Guida autonoma. http://www.treccani.it/vocabolario/guida-autonoma_%28Neologismi%29/, 2017.

[55] Ucsusa.com. Self-driving cars explained. https://www.ucsusa.org/clean-vehicles/how-self-driving-cars-work, 02 2018.

[56] Avinash Sharma V. Understanding activation functions in neural networks. https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0, 03 2017.