POLITECNICO DI MILANO

School of Industrial and Information Engineering
Master of Science in Biomedical Engineering

Deep Convolutional Neural Networks for real-time analysis of human emotions to extend robotic empathy

Supervisor: Pietro Cerveri
Co-Supervisor: Alessio Spina

Thesis of:

Niccolò Cerrone
ID: 905382

Matteo Corti
ID: 899305

Academic Year 2018-2019

Acknowledgments

This Master Thesis work has been carried out in collaboration with Softec s.p.a, during an eight-month long internship. The Softec company provided us all the products, devices, competences and the location we needed to complete this work.

Thanks to Alessio Spina, our co-tutor, all the people in Orchestra team and all the other colleagues who collaborated with us, providing us technical support, supervising our work and making out time in the work place unforgettable.

In particular, thanks to our thesis supervisor, Professor Pietro Cerveri, for the chance to work at this interesting and innovative project. Thanks for following us throughout this path, providing us with the best competences and advises and for making us feeling like engineers, not only students.

Thanks to our families for inciting and accompany us in our entire life, for guaranteeing our studies and because the results we've obtained are due to your sacrifices and support.

Thanks to our friends and our mates, who walked with us up until now and shared our emotions and lives. Thanks to those who made this difficult route more comfortable and enjoyable.

We also would like to thank everyone who helped in collecting the photos and who took part who took part of the project by helping us collecting the photos we needed so the experiment would reach a successful outcome.

Finally thanks to Politecnico di Milano, the place where we spent the most of the recent years, where we learned, suffered, laughed, met new people and the place that got us out there, allowing us to become Engineers.

Sommario

L'intelligenza artificiale, un ramo della scienza informatica, sta diventando sempre piú popolare nella vita di tuti i giorni, grazie alla possibilitá di condividere e gestire facilmente una grande quantitá di dati. Questa tecnologia permette l'implementazione di sistemi in grado di eseguire compiti legati all'intelligenza umana, simulando ragionamenti e abilitá umane. Di recente, il forte incremento nella disponibilitá dei dati e nello sviluppo di sistemi con una piú ampia potenza di calcolo ha favorito l'utilizzo di tecniche di deep learning. Di conseguenza, come si vede in letteratura, una gran quantitá di applicazioni basate sull'AI é stata sviluppata.

In questo progetto il lavoro é focalizzato su una delle piú particolari abilitá del cervello umano, il riconoscimento dell'espressioni facciali. Il riconoscimneto facciale é una funzione che si puó spesso ritrovare in molti strumenti SW ormai da anni in commercio, come per esempio su dispositivi come smartphone, tablet e PC. Seppur giá largamente sfruttata nell'ambito della tecnologia, l'analisi delle emozioni umane sta diventando sempre piú diffuso, specilamente nel settore sanitario, nel settore commerciale, in ambiti di sicurezza, ingegneristico e anche nella robotica.

Per questa ragione, l'obbiettivo del nostro progetto é proprio quello di combinare il potenziale offerto dall'AI con quello offerto dalle nuove tecnologie robotiche, sviluppando un sistema autonomo in grado di migliorare le doti empatiche di un robot umanoide denominato Pepper. Pepper é un robot con caratteristiche semi-umane e lo scopo principale é quello di gestire le sue abilitá comunicative grazie allo studio delle emozioni del suo interlocutore.

Il riconoscimento delle emozioni facciali avviene tramite l'elaborazione dalle immagini, il che é un dato particolarmente importante poiché non richiede attrezzature particolarmente complesse e permette di acquisire i dati molto facilmente; é sufficiente infatti il solo utilizzo di una telecamera senza creare inconvenienti al soggetto.

Un fatto che viene spesso trascurato, sebbene di particolare importanza, é che le emozioni non possono essere divise in categorie ben definite, ma quando parliamo di emozioni facciali il volto puó indicare emozioni diverse se visto da persone differenti. Le emozioni sono una caratteristica talmente soggettiva che la stessa identica faccia puó indurre a risposte diverse sia se vista da persone diverse, sia se vista dalla stessa persona in momenti diversi.

Grazie all'utilizzo delle reti neurali specificamente costruite ed efficacemente addestrate é possibile analizzare le immagine ed estrarne dati relatvi allo stato emozionale dei soggetti inquadrati.

La problematica piú grande nello sviluppare un'efficiente rete neurale di questo tipo é quella di trovare il dataset piú adatto per addestrare in maniera corretta il modello, evitando errori di adattamento per garantire buone predizioni.

Nonostante questo sembri essere un progetto molto arduo, si é deciso di sfruttare le migliori tecniche di deep learning per sviluppare un sistema per il riconoscimento delle emozioni facciali che potesse essere eseguito in real-time e che potesse essere integrato con una tecnologia robotica umanoide.

Anche se alcuni dataset giá predisposti sono disponibili online, la scelta é ricaduta nel

creare un nostro dataset partendo da zero, sfruttando il fatto di poterlo creare a seconda delle nostre esigenze ed associando ad ogni immagine il label corretto.

La nostra scelta é stata quella di considerare 7 emozioni complessivamente: le 6 principali (*felicitá*, *tristezza*, *paura*, *rabbia*, *sorpresa* and *disgusto*) con l'aggiunta dell'espressione *neutrale*.

Una volta selezionate le emozione adatte e completato il dataset, quest'ultimo é stato modificato per ottenere 4 diverse composizioni di dataset partendo da quello originale; in questo modo abbiamo avuto la possibilitá di comparare i risultati ottenuti e di scegliere quale garantisse un maggior successo nel raggiungere l'obiettivo finale.

L'approccio utilizzato é simile anche per la realizzazione delle reti neurali: 4 differenti strutture sono state create ed ognuna di queste é stata addestrata con ciascuno dei dataset sopra descritti. Testando tutte le diverse strutture di rete neurale con ogni configurazione di dataset abbiamo trovato la miglior combinazione che ci permettesse di completare al meglio il progetto.

Durante la creazione della rete abbiamo dovuto tenere in considerazione il fatto che il modello, una volta completato l'addestramento, dovesse essere processato da un dispositivo embedded. La gestione dell'analisi emozionale, infatti, avviene grazie all'utilizzo di un dispositivo chiamato Google Coral Dev Board, una scheda basata creata da Google che sfrutta l'utilizzo di una nuova tecnologia chiamata tensor processing unit (TPU). La TPU é utilizzata per processare la rete neurale e quindi velocizzare la valutazione delle emozioni del soggetto grazie all'analisi di un flusso di immagini provenienti dalla videocamera del robot. Per poter sfruttare a pieno questa nuova tecnologia é necessario convertire la rete, strutturata grazie all'utilizzo di Tensorflow, in una versione piú rapida e leggera utilizzando TF lite. L'utilizzo di questo dispositivo esterno é assolutamete fondamentale dato che la componenti hardware incluse all'interno del robot non sono adatte ad eseguire questo tipo di operazioni ed é necessario il trasferimento di tutte le operazioni computazionali.

Nel frattempo abbiamo sviluppato la parte principale del codice, sfruttando le potenzialitá di alcune librerie specifiche del linguaggio Python e per ottenere immagini adatte ad essere processate dalla rete neurale é stato necessario lo sviluppo di un programma di elaborazione video. Questo programma, installato sul dispositivo, consente di catturare fotogrammi provenienti dalla videocamera, rilevare e codificare i volti delle persone nel video e analizzare il genere e l'etá di ogni persona. Questo processo iniziale é essenziale per estrarre tutte le caratteristiche (ID, etá e genere) delle persone che il robot sta osservando, con cui interagisce e anche per preparare tutti i frame utili per alimentare la rete e ottenere dati relativi alle emozioni del soggetto.

Questo dispositivo é infine collegato ad un server appositamente creato che ha il compito di gestire tutti i dati provenienti dalla Coral Dev Board. Il server é poi connesso da un lato con un database dove tutti i dati vengono salvati, mentre dall'altro é in comunicazione con il robot, al quale deve fornire i dati necessari quando richiesti.

Lo sviluppo del lavoro é stato suddiviso suddiviso in varie fasi.
Per iniziare, svariati tentativi per addestrare le differenti reti neurali sono stati eseguiti utilizzando le diverse composizioni di dataset. Abbiamo quindi trovato che il migliore dataset e la miglior configurazione di rete neurale in grado di fornire la predizione emozionale piú accurata possibile mostrasse un valore di accuratezza molto vicino al 55%: risultato accettabile, seppure non ottimale per la gestione in tempo reale delle reazioni del robot. Con molta probabilitá questo risultato é legato al fatto di avere addestrato il modello utilizzando un dataset solo parzialmente completo, mentre utilizzandone uno piú ampio i valori di accuratezza non potrebbero che crescere.

Successivamente, la fase di preelbaorazione delle immagini e i processi di riconoscimento

delle persone sono stati sviluppati per estrarre tutti i dati d'interesse e una volta che sia questa parte di elaborazione, sia il modello adeguatamente convertito sono stati installati sul dispositivo, l'intero sistema era pronto per analizzare il video e fornire tutto ció di cui Pepper avesse bisogno.

Alla fine il sistema é collegato al server che permette di trasferire i dati al database (MongoDB) e al tempo stesso di estarre il parametro di cui Pepper ha bisogno. Il robot é implementato tramite l'utilizzo del "Behaviour Motor System" che gli permette di richiedere al server i dati di cui ha bisogno per gestire il suo comportamento in base allo stato emozionale momentaneo del soggetto.

Eseguendo l'intero programma abbiamo notato che le prestazioni ottenute sulla scheda non sono del tutto soddisfacenti, infatti questa configurazione si rivela eccellente per la gestione e l'analisi dei dati ma non sufficientemente veloce per gestire una conversazione in tempo reale. Per risolvere questo problema sono state apportate alcune modifiche al robot BMS.

In conclusione ció che abbiamo ottenuto é un sistema integrato "robot/dispositivo" in grado di profilare e riconoscere le emozioni delle persone. Questo non é certamente perfetto in termini di prestazioni e velocitá, ma potrebbe essere un buon compromesso per quanto riguarda l'analisi e la definizione dei dati.

Abstract

Artificial Intelligence, a branch of Computer Science, is becoming more and more popular in the everyday life, thanks to the possibility of sharing a large amount of data and managing them. Artificial Intelligence allows the implementation of systems that can execute tasks associated to human intelligence, simulating human skills and reasoning. Recently, the increase in available data and computing power has promoted the use of deep learning techniques. Therefore, a huge amount of AI applications exist, as it can be seen in literature.

In this project, the work is focused in one of the most particular ability of the human brain, the Facial Expression Recognition (FER). Face Recognition can be often found in commercial tools, applied on devices like smartphones, tablets and PC. Although already largely used in the technology sector, people emotion analysis is becoming more widespread, especially in healthcare, in marketing and business, in security, software engineering and even in robotics.

For this reason, what we want to do within our project is precisely to combine the potential offered by AI and those of robotics. We want to develop an autonomous system capable of improving the empathetic qualities of a humanoid robot: Pepper. Pepper is, in fact, a robot with semi-human features and our goal is to be able to manage its communicative aspect thanks to the study of the interlocutor's emotions.

Facial expression recognition from images is particularly important because it does not require advanced equipment and allows to easily acquire data with a camera without inconveniencing the subjects.

A fact that is not usually addressed, although particularly important, is that emotions cannot be divided into strict classes. Indeed, when taking about facial emotions, the same face could be recognized as expressing different emotions if seen by different people. Emotions are such a subjective aspect that the exact same face can induce different answers if seen by different people or even if it is seen by the same person in two distinct moments.

With an Artificial neural network, specifically edited and efficiently trained, it could be possible to analyze and extract emotional data from an image. Some of the issues in doing this include finding the best parameters and weights to assign to the network, since they are a lot and often hardly solvable.

The most arduous challenge in developing an efficient Neural Network is indeed to find the most efficient dataset suitable to properly train the Network avoiding bad fitting of the model to guarantee good predictions.

Although this proves to be a really challenging project, we tried to exploit the best deep learning techniques to develop a system for Facial Emotion Recognition that could run in real time and could be integrated in an humanoid robotic technology.

Even if some already created datasets were available, we decided to create the entire dataset from scratch, taking advantage of the fact that it would have been possible to choose how many and which emotions analyze. In this way is also possible to associate the best label to each photo, sorting the entire dataset to re-edit them in the most suitable manner to best

feed the net.

The choice is to process 7 emotion in total: the 6 primary emotion (*joy*, *sadness*, *fear*, *anger*, *surprise* and *disgust*) with the addition of a *neutral* state.

Once the emotions were selected and the dataset completed, this was elaborated in order to get four different compositions starting from the original one. So to have the chance to compare the different results obtained by each dataset and finally decide which one could better perform and reach the final goal.

A similar approach was adopted with Neural networks: four different configurations were tested. Combining all the different CNN structures with each of the possible datasets, we found the best model to be used for finalizing the project.

While creating the neural network we took into consideration the fact that the model, once trained, should have been processed by an embedded device. The management of the emotion analysis is done trough a device called Google Coral Dev Board, a TPU-based board created by Google that takes advantage of the TPU technology. This TPU is used for elaborating the CNN in substitution of CPU and thus to speed up the evaluation of the emotion of the subject from a flux of frames coming from the camera of robot. To exploit this new technology it was necessary to convert the Tensorflow Network in its Lite version, using a specific interpreter installed on the device to run the whole process. Utilizing this additional device is a fundamental step since the hardware component of the robot is not designed to do this kind of operations and the transfer of all the computational operations is necessary.

Meanwhile the main part of the code is developed, using the potentiality of some specific libraries of Python language. In order to obtain images suitable to be processed by the neural network, the development of a video processing program was necessary. This program, installed on the device, allows to capture frames coming from the camera, detect and encode the people faces in the video and analyse each person gender and age. This initial process is essential to extract all the characteristics (ID, age and gender) of the people the robot is looking at and which are interacting with it. This is also necessary to prepare all the frames for feeding the Network and obtain emotional data.

The device is then connected to a server, specifically created, which has the role to manage all the data coming from the Coral Dev. This server is then connected on the one hand with a database where all the data are stored, while on the other is linked to the robot where its aim is to extract the necessary data when Pepper needs them to manage the conversation.

The development of this work is quite complex and consists in more phases.

Just to start, several attempts to train different neural networks have been made. Different configurations to be analyzed have been created and various datasets have been used for the training phase. What was found out is the best dataset and the best Neural Network's configuration that allow the most precise emotions prediction possible. What was obtained is an acceptable prediction net, about 55% of accuracy, but not an optimal choice to manage with a real time emotion recognition. However, the fact that the dataset used for the training is only partially completed, due to the decision of creating a new one from nothing, we suppose this result can only increase completing the dataset.

After that, the preprocessing phase and people recognition processes are developed in order to feed the network and collect all the people's data we're interested in. Then this part, together with the suitably converted model, is installed on the Google coral Dev Board to execute the whole path to extract the data.

Finally, the system is linked to a server module which allows the data transfer to a database (MongoDB) and at the same time it allows to pick the stored data, according with Pepper's requests. Pepper is implemented by the Behaviour Motor System in order to request the data regarding its interlocutor, modifying its behaviours and speeches depending on the

type of data it receives.

Running the code we noticed that the performances obtain on the board are not totally satisfactory. This configuration proves to be excellent for data management and analysis but not sufficiently fast to manage a real time conversation. To solve this issue some adjustment have been made in the robot BMS.

At the end, what we have obtained is an integrated "device-to-robot system" able to profile and recognize emotion just focusing on a person. This system is, certainly, not perfect in terms of performances and speed, but it could be a good compromise as regards people data analysis and outlining.

Contents

List of Figures

List of Tables

List of abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
ASIC	Application Specific Integrated Chip
BMS	Behaviour Management System
CM	Confusion matrix
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
GCDB	Google Coral Dev Board
FER	Facial Expression Recognition
HTTP	Hyper Text Transfer Protocol
MFD	Modified Free Dataset
MLD	Modified Landmark Dataset
ML	Machine Learning
NN	Neural Network
RFD	Raw Free Dataset
RLD	Raw Landmark Dataset
RSO	Robotic Service Orchestration
TF	TensorFlow
TPU	Tensor Processing Unit

Chapter 1

Introduction

1.1 Context

In recent years the concept of artificial intelligence (AI) is becoming increasingly popular and more and more are the possible fields in which it is becoming of primary importance. The foundations for this AI concept were laid in the decade between 1950 and 1960 and since then its development has continued to grow, with an exponential improvement with the arrival of the new millennium, where the sharing of large amounts of data promoted a better use of this technology [1].
To give a definition of AI, it is a subject of Computer Science aimed at building machines and computers that can enhance logical operations. It can therefore be generally defined as "the discipline that studies the design, development and implementation of systems capable of simulating human skills, reasoning and behavior". In conclusion it includes all systems which have the ability to execute tasks naturally associated with human intelligence, like speech recognition, decision-making, visual perception and translating languages.

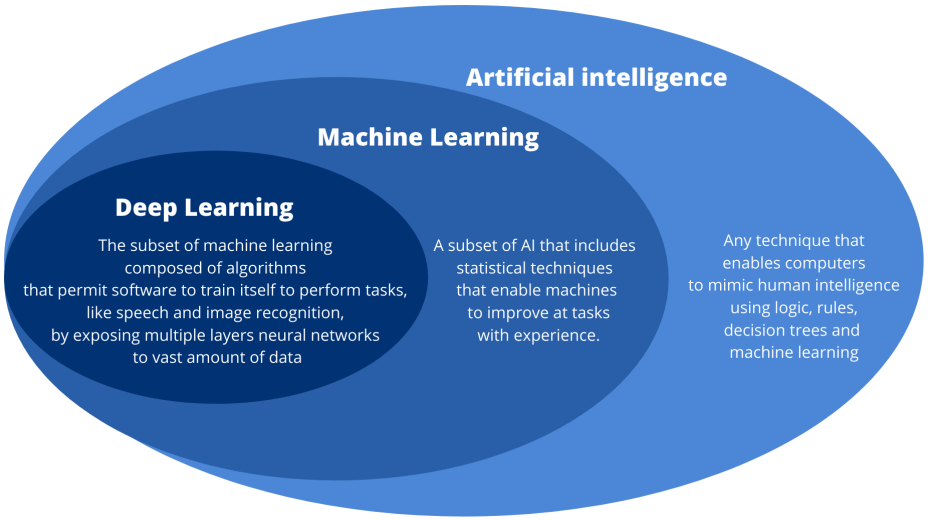

Figure 1.1: *Artificial Intelligence Sub-division*

As we can see from the above figure, Machine Learning (ML) is a subset of AI. The aspect that separates ML from the rest of Computer Science factors is its extraordinary ability to modify itself when exposed to more data and it does not require human intervention to make certain changes, so that machines can observe, analyse and learn from data and mistakes just like our human brains can.

From figure 1.1 it is also clear that Deep Learning (DL) in turn, is a sub-field of machine learning and precisely it is one of the most powerful and fastest growing applications of AI. DL is used to solve problems which were previously considered too complicated and involve a very large amount of data. It occurs through the use of neural networks, which are layered to recognize patterns and complex relationships in data. The application of DL requires a vast dataset and mighty computational power to work. Recently, due to an increase in the ready availability of computational power and increasingly large training databases to work with, the machine learning techniques of neural networks has seen resurgence in popularity.

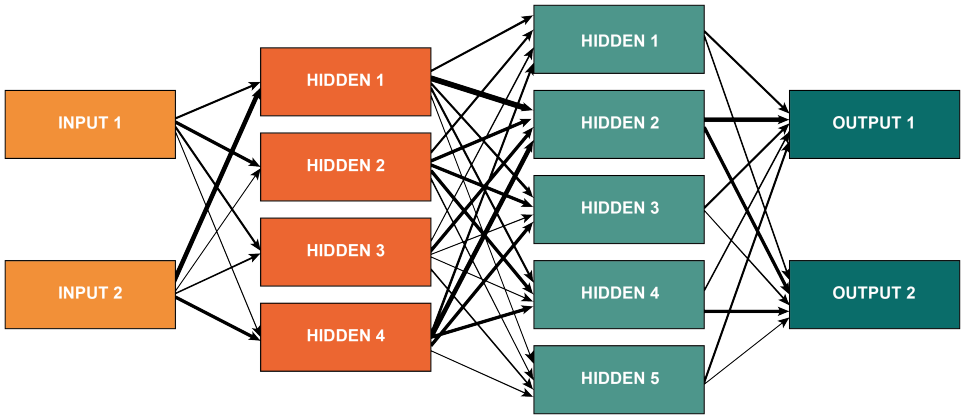

Figure 1.2: *Example of an Artificial Neural Network with multiple hidden layer*

The main difference between deep learning and machine learning is due to the way data is presented in the system. Machine learning algorithms almost always require structured data, while deep learning networks rely on the "hidden" layers of NN.

Machine learning algorithms are designed to "learn" to act by understanding labeled data and then use it to produce new results with more datasets. However, when the result is incorrect, there is the need to "re-teach" the classifiers.

Deep learning networks do not require human intervention, as multilevel layers in neural networks place data in a hierarchy of different concepts, which ultimately learn from their own mistakes. However, even they can be wrong if the data quality is not good enough. The main differences between ML and DL are:

1. Because machine learning algorithms require bulleted data, they are not suitable for solving complex queries that involve a huge amount of data.

2. The real use of deep learning neural networks is on a much larger scale. In fact, given the number of layers, hierarchies, and concepts that these networks handle, deep learning is suitable for performing complex calculations, but is not required for simple ones.

3. Both of these subsets of AI are somehow connected to data, which makes it possible to represent a certain form of 'intelligence'. However, deep learning requires much more data than a traditional machine learning algorithm.

4. Machine learning algorithms, on the other hand, are capable of learning by already programmed criteria. The reason for this is that deep learning networks can identify different elements in neural network layers only when more than a million data points interact.

Nowadays AI is of primary important in a vary large number of field such as sales, marketing, cyber-crime, risk analysis, public security and healthcare and in each of these it assumes a key role. This technology is influencing consumer products and has led to significant breakthroughs in healthcare and physics [2] as well as altered industries as diverse as manufacturing, finance and retail [3].

Regarding healthcare, many companies are investing in and researching ways in which artificial intelligence can help improving our system. Many are the fields in the sanitary system which, benefiting from this innovation, have seen a marked improvement. Some examples are personalised drug protocols, innovative diagnostic tools, robotic device to assist in surgery and even more complicated systems able to work independently. The improvement introduced by AI in the health system is evolving not only invasive techniques such as those mentioned above, but also various non-invasive techniques have found a strong increase in reliability and popularity [4].

By widening our gaze on the general picture, we can see that, even without entering very complex areas, AI is part of everyday life for a large part of the modern population. In fact, we use AI in many tasks of everyday life; for example, voice recognition in the increasingly popular home automation systems, face recognition now present in most smartphones, email filtering, the predictive search on the web browser or the search of the best route in driving directions. These make us understand the importance of AI in our life and the possibility to improve the applications, as in our case, to recognize the emotional state of people.

Facial expressions play a very important role in recognition of emotions and are used in the process of non-verbal communication, which has a really big impact on every day life; therefore they are fundamental in daily emotional communication, just next to voice tone. What makes facial expressions of particular interest is that although they can be voluntarily generated by means of decisions established by the subject, in most cases they are unconscious reactions to unexpected stimuli and arise naturally on people's faces. That means firstly that they are often not controlled by subject will and precisely because of their naturalness they are genuine indicator of the emotional state of the people. Finally they are an indicator of unconscious feelings, allowing a man to silently express themselves.

As a consequence, studies and research based on Facial Expression Recognition (FER) are becoming more and more widespread as the analysis of this data provides valuable information applicable in countless sectors. This causes that its popularity arises from really vast areas of possible applications, such as:

- diagnosis and study of diseases [5] [6][7][8]
- driver fatigue detection [9]
- security applications [10][11][12]
- socially intelligent robots [13]
- software engineering [14]
- education and gaming [15][16]

Given the large number of possible interested parties in FER and its recent popularity, the most modern approaches are used to allow machines to automatically perform this task and DL is the technique which better succeed in this. Images or videos extracted from one or more cameras are passed through a neural network in order to perform an automatic recognition of facial expression. What is the base of all the DL algorithms for FER is the division in 3 main step [17]:

- Face detection
- Feature extraction
- Expression classification

Going deeper into the analysis of the individual steps that compose the RES, we note this is a particularly complex process, as it represents the sequence of single complicated operations.

Starting from the first step we can note that face detection is already very complex itself [18] as it requires the use of Computer Vision (CV), a sub-field of artificial intelligence. The term CV refers to the process in which the goal is to build a computer that replicates the visual intelligence of the human brain, which is the phenomenon that makes machines able to see the environment as humans see it. Thanks to this technology, always based on the use of neural networks and on their training, it is possible to identify the presence of the face of one or more people within images or video frames and proceed from there.

Feature extraction, meanwhile, is the name for methods that select and combine variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original dataset [19].

This process is very useful when you need to reduce the number of resources needed for processing without losing important or relevant information. Thanks to this operation the minimization of the data and the machine's efforts facilitate the speed of learning and generalization steps in the ML process leading to accuracy improvement and over-fitting risk reduction [20].

For what concerns the last point, expression classification can be considered the most problematic one, and although great improvements have been made in the field of AI, and more specifically in DL, the recognition of facial expressions through machines like computer is still hard to be performed. What is quite typical using DL techniques is that the analysis of data is performed in order to distinguish objects, animals, humans or in general well defined categories, but that is not what happens for FER. Facial expressions are usually not the outsourcing of one single emotion but the sum of multiple contemporary feelings, which does not allow the association of an expression with a single strict category. This makes their prediction very complicated and more and more different approaches, mainly in the choice of training dataset, were tried in order to solve this problem.

Looking at the whole process of FER, there are also several factors which can influence the final result and that we must keep in consideration [21]. Illumination is one of the most challenging problem since variation in light conditions can cause errors both in the identification of the face and, if the first task is performed correctly, in the expression recognition. Face pose is another heavy component, given that variation in face position changes the spatial relations among facial highlights and causes serious distortion on the FER analysis. Two more important factors are occlusion and clutter background since the impact of environments and backgrounds around individual can't be neglected and the coverage of part of the face reduce significantly the precision of the prediction.

Among the factors just expressed, light condition and head position changes are the main factors that affect the quality of emotion recognition systems using cameras. Especially sensitive for these factors are methods based on 2D image analysis, while methods in which 3D face models are implemented are far more promising and seems to be less affected by these types of errors [22].

1.2 Proposed project

In this context, a lot of Deep Learning applications have, recently, been developed and have shown excellent results, both in facial and emotion recognition. However we need to keep in mind that in FER process, the way in which an 'emotion' is interpreted is usually subjective and not easily recognisable.

The aim of our project is to recognize seven basic emotional states: neutral, joy, surprise, anger, sadness, fear and disgust based on facial expressions, to perform this classification using imaging extracted from a real time acquisition camera and to use the obtained information for two main reasons: to manage the "behaviors" of a humanoid robot and to profile various interlocutors.

Figure 1.3: *The seven emotions, from top left to bottom right: joy, surprise, anger, sadness, fear, disgust, neutral*

To do this, we need to create a Deep Neural architecture [23], which is the best approach in terms of data and computation. As previously exposed, Deep Neural Network allows to analyze and extract the features of each image used to train the Net, to store them inside the network and to recognize that features from each new image given as input, associating it to one of the seven emotions. A fundamental step to obtain a good prediction is the choice of the datasets used to train and validate the Neural Model and despite big amount of datasets containing images of the face with a corresponding expression label are present online, for the needs of our project we decided to create a new dataset from scratch.

The development of a neural network for the recognition of facial emotions, although not simple at all, is not in itself an innovative idea since in recent years various attempts, with more or less satisfactory results, have been examined and described in the literature. The innovation brought by our project is the development of an autonomous system, which associates the skills of DNN with a robotic intelligence. The union of these two generates an ideal platform for robot entertainment and subject profiling. The interaction between an humanoid robot and a human subject allows us to take full advantage of our system: the neural network analyzes in real time the images of a video camera and generates feedback on the emotional state of the subject which allows an adequate reaction by the robot.

This system exploits a fairly recent technology produced by Google, the Google Coral Dev Board, a board which performs high-speed ML inferencing equipped, in addition to the usual CPU and GPU, with an on-board Edge TPU coprocessor capable of performing 4 trillion operations per second 1.4. The Edge TPU is a small Application Specific Integrated Chip (ASIC) designed to offer a high-performance machine learning interpretation and thanks to the small size and low power consumption make it perfect for integration into IoT hardware products for image recognition

Figure 1.4: *Google Coral Dev Board*

For what concerns the robotic intelligence, we used Pepper, figure 1.5, a humanoid robot granted by the Softec Spa company where, in collaboration with Politecnico di Milano, this study was carried out.

1.2.1 Expected results

The importance of a stand alone system is the linking between a robot and its "intelligence", so that the robot is the effective actuator of the aims, without overloading the hardware, allowing to reduce the amount of data stored and processed by the processing unit of the robot, permitting to reduce the computational time of the DNN and in general of all the FER algorithms and well replacing a powerful machine, such as a computer, saving computational space and money.

Figure 1.5: *Pepper*

The expected objectives are:

1. Development of CNN able to perform adequate predictions
2. Development of an algorithm for real time processing of visual information
3. Development of an advanced stand alone system optimized to FER analysis
4. Storage of the results in a database for profiling subjects interacting with the robot
5. Increasing Pepper's communicative ability and empathetic skills

Chapter 2

State of the art

In the last years, many technologies based on AI have been developed in order to classify human emotion, using a wide range of different approaches, starting from different kind of input, passing through variations in pre-processing phase until the variability in the conformation of the neural network itself.

According to the literature the main approach is related to visual information, even if it is not the only possible type of input data since in few cases auditory datasets were used [24][25].

For what concerns our project we are going to analyse those articles regarding neural networks able to recognise human emotions expressed by facial expression [18].

Most of the considered classifiers were based on still images, either in color or in gray-scale [26][27][28], while a restricted number of examples were based on videos [29][30], infrared images [31][32] or 3D data [33][34] in order to avoid some limitations present in still frame (variation of light and relative position of the subject).
What matches almost all these works is the presence of a very similar pipeline:

- Selection of adequate dataset
- Preprocessing of the visual information
- Development of a neural network for classification of the emotions

The choice of the dataset is a critical variable which can determinate the degree of reliability of the final results and precisely for this reason many different approaches where considered.

Facial expressions are a very particular case and cannot be considered as strictly belonging to a single specif category since each one is the manifestation of combined different emotions. Moreover, facial expressions are actually ambiguous because, beyond their manifestation, even their interpretation in not an easy task, since the same expression can be interpreted as being representation of different emotions if seen by different viewers or if seen from the same person in different moments. This is the reason why some authors decided to based their works on emotions considered as combination of various basic ones [35][33] or to evaluate images with multiple label[36] taking into account opinion of different viewers in order to consider the probability of the same expression to belong to different classes.

Despite what just described, the most common approach in literature was to evaluate the expressions as belonging to a single specific category. The largest number of methods were based on six basic emotions (*anger*, *disgust*, *fear*, *happiness*, *sadness* and *surprise*) [37][38], sometimes including *neutral* class [39] and in few cases *contempt* was considered.

When the dataset was not sufficiently large, some ulterior considerations were done: the first one was to apply some non invasive transformation to increase the number of examples (data augmentation) [40], while the second one is to use a pretrained neural network on a bigger dataset concerning a similar task and then specialise it with the small useful face emotion recognition dataset[41].

One more step developed in order to increase the ability in classifying emotions was the preprocessing of the images in order to focus all the computation power on the data useful for the effective classification. Preprocessing can be considered as the sequence of operations to obtain an image easier to be elaborated by the classifier. This sequence typically starts from the face detection, the localization of point of particular interest, such as eyes and mouth, for the alignment and the pose correction assuming that all the images were taken from the same angulation [42] [43]. In some other cases also brightness normalization [44][45] and dimension adjustment were performed. Most of the classifiers work on still RGB or gray scale images [46][37][38], while others use videos [36][47], to evaluate the act of changing emotion. As we can see, although the preprocessing is various, its aim is always to reduce the information content of the input data in order to focus all the computational power on data useful for emotion recognition.

As shown before the development of a neural network is the last step of the common pipeline and it's usually divide into two different sub-points:

- Feature extraction using a Deep Neural Network(DNN)

- Classification based on extracted features

Most of the times the DNNs used for the feature extraction are Convolutional Neural Networks(CNN) [28][27], but are also reported examples in which Deep Belief Networks [37], Deep AutoEncoders [39] and Recurrent Neural Networks [48] are adopted. Only after the extraction of the feature the effective classifier can be trained.

Sometimes the two phases described just above are connected and the training is performed on the full network without distinguishing the classifier from the feature extractor [49][50]. A softmax layer is often used to perform a classification using such a network configuration [49][39][51], but other methodologies such as Deep Neural Forests [50] or the exploitation of Support Vector Machines [52][53] can be utilized.

On the contrary, reading the FER for a humanoid robot is not usual, although more and more applications on robots are developed to improve their ability to perceive the external environment, including their own body, by utilizing multiple sources of sensory information, such as vision, audition, and object manipulation [54]. The trial in combining multiple sensorial acquisitions to create a good model which replicates human behaviour to use the robot in every day life is the start point in most of the researches. For example, feed-forward neural networks can be used respectively implement both upper and lower facial Action Units analysers to recognise six upper and 11 lower facial actions including Inner and Outer Brow Raiser, Lid Tightener, Lip Corner Puller, Upper Lip Raiser, Nose Wrinkler, Mouth Stretch etc. An artificial neural network based facial emotion recogniser is subsequently used to accept the derived 17 Action Units as inputs to decode neutral and six basic emotions from facial expressions. Moreover, in order to advise the robot to make appropriate responses based on the detected affective facial behaviours [44], Latent Semantic Analysis is used to focus on underlying semantic structures of the data and go beyond linguistic restrictions to identify topics embedded in the users' conversations. The overall development is integrated with a modern humanoid robot platform under its Linux C++ SDKs [55].

Chapter 3

Materials and Methods

The main goal we wish to achieve in our thesis project is to develop a system able to recognize facial emotions and to integrate it on a humanoid robot to provide:

- Real time processing of visual information for the management of robot abilities in the interaction with people
- Analysis of data for profiling subjects interacting with the robot

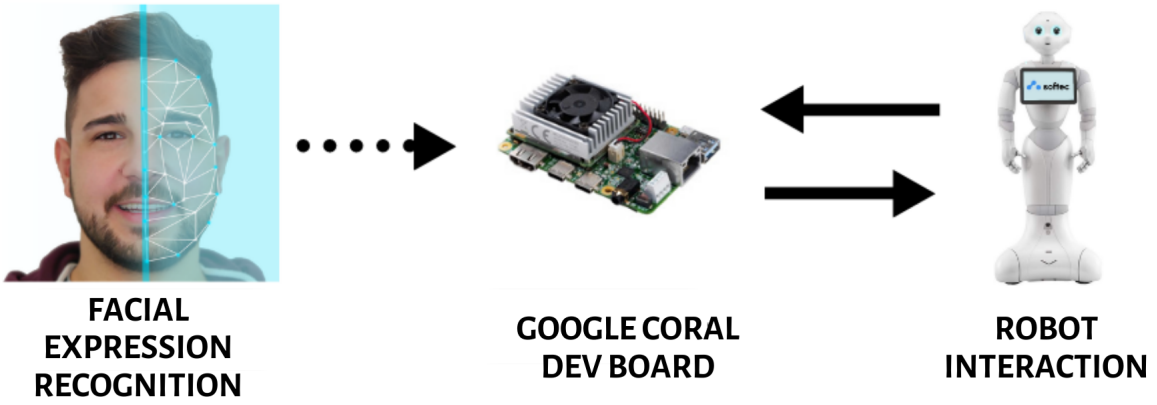

Figure 3.1: *General representation of proposed project*

In order to reach the target, we followed the diagram reported in figure 3.1 where the three main phases are shown:

- Creation of a dataset and development of a CNN adequate for FER
- Development of a software for real time image acquisition, running on Google Coral Dev Board to exploit the computing power of the Edge TPU mounted on it for faster emotions analysis
- Effective communication between our system and the robot.

The arrows in both direction show that the robot is the first actor of the process since it provides the video streaming that will be analysed and it is also the end-effector since it receives information about the emotional state of its interlocutor and provides different actions according to what the human is feeling.

Python3

All the algorithms we used for our project are built using Python[1] as programming language. Python is an interpreted, high-level, general-purpose programming language whose main features are its the dynamism, simplicity and flexibility. Although Python is considered an 'interpreted language', the source code is not directly converted in machine language. The main advantage of this language is its versatility, being able to move with ease in different pre-compiled libraries and execute a big number of functions. When it comes to its power, Python's performances are comparable or higher than other interpreted languages, but not as performing in computational time and math calculation.

It is developed under an OSI-approved open source license, making it freely usable, open to distribution and even for commercial use.
Python's library useful for the project will be discussed in the next sections.

3.1 Facial Expression Recognition

The first step is the development of a system able to recognize the expression manifested on people's faces. The easiest and most efficient way to do so is through the use of a network based on DL algorithm; in this specific case a CNN was developed, as described graphically in figure 3.2.

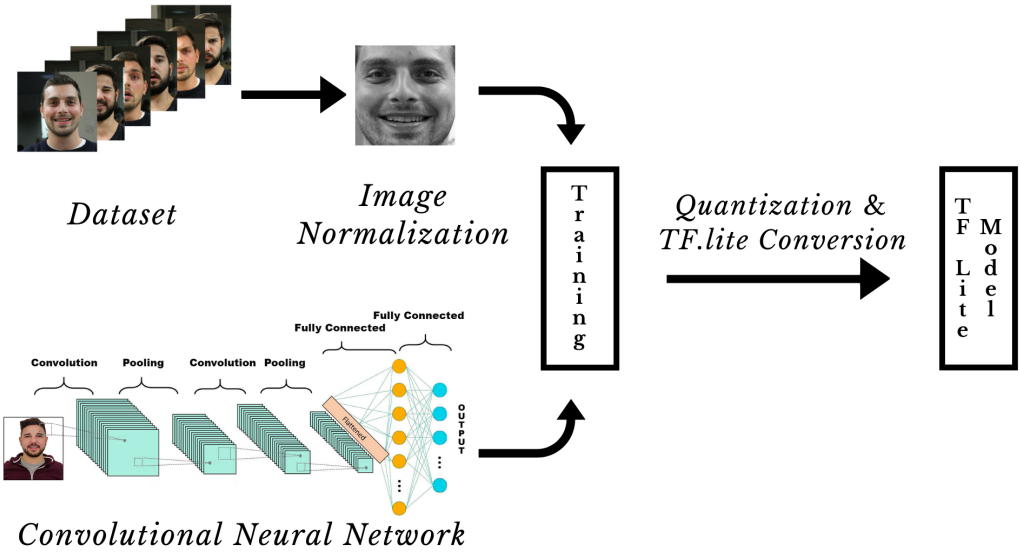

Figure 3.2: *Passages to create a FER model*

This pipeline shows all the steps that have been taken in order to be able to create a neural network model and to complete the first part of the project. Starting from the Dataset, passing through images preprocessing, the creation of the actual network, up to the training and finally the conversion phase of the CNN in Tensorflow lite. This picture highlights all the phases of the facial expression recognition that will be discussed in the following paragraphs.

[1] `https://www.python.org`

3.1.1 Dataset

One of the most important components to work on and that can seriously affect the obtained results is the choice of the dataset used to train the network. Both in terms of completeness and choice of the recognizable categories. As already reported in the literature, the choice of emotions to be recognized can be multiple and can vary according to needs. As far as our project is concerned, we have chosen to base the analysis on only primary emotions (*joy*, *sadness*, *fear*, *anger*, *surprise* and *disgust*) with the addition of a *neutral* state. The motivation that justifies this addition is that in most cases those who interface with the robot do not show a particular expression for the entire duration of the interaction, but the presence of recognizable emotions is only found occasionally, interspersed with longer periods of neutrality.

Once the first step was over and the categories of interest were selected, it was necessary to find an adequate dataset to train the network that could provide acceptable results. Although some datasets, with the same seven selected emotions, were possible to be found online, we preferred creating a new one, starting from scratch, for two main reasons. The first one is the poor reliability of the online datasets, as there were many cases in which the label did not coincide with the respective images; and the second, and even more important, is the need to develop a system for commercial purposes which made the online research almost totally unsuccessful. Considering this choice, we tried to keep in mind three key concepts for creating a good dataset:

- The dataset should be heterogeneous
- The categories should preferably be balanced
- The number of samples should be consistent

Going more into details, if we talk about heterogeneity we can see that, to obtain a satisfactory dataset, people of all genders, ages and nationalities should be included. For what concerns origins and ages, it was not possible to obtain a number of uniform samples as the subjects are mostly Caucasian and aged in a range between 20 and 35 years. This data is the consequence of our social context since mainly relatives, friends, colleagues and university students have been considered. Contrarily, no special consideration about gender are highlighted since analysed samples were heterogeneous.

When it comes to balance among classes, we refer to the number of images contained in each of them. The aim is to obtain the same, or at least very similar, number of photos in all the considered categories to ensure homogeneous learning without privileging one emotion over the others. So, to keep this balance, we performed a protocol about the way in which the photos were taken: we asked every single subject to express all the seven required emotions in sequence as follow:

- *joy*
- *sadness*
- *fear*
- *anger*
- *surprise*
- *disgust*

- *neutral*

This way we obtained a first raw dataset with an almost equal number of samples for all the emotions.

This means that the expressions captured in the photographs do not represent a real emotional state, but they are the externalization of a forced emotion and, consequently, they do not reflect a natural involvement of the subject. This may therefore indicate an incomplete correspondence between the emotions that the subject normally expresses and those reported in our dataset.

Since our dataset was made up of simulated expressions that may partially differ from reality, we decided to check each single image and to discard those which, in our opinion, were strongly out of context. Keeping in mind that not all emotions can be easily reproduced on command and considering that every single subject has its own way of expressing them, we tried to intervene in way that wasn't too invasive on the starting dataset; for this reason we deleted only those cases where the externalization of the emotion was, from our perspective, totally wrong.

We proceeded by creating two datasets: one based only on the skills of each interviewed individual to express emotions on request and a second one where an accurate sorting was performed by us to verify the true correspondence between the emotion requested and the one actually expressed.

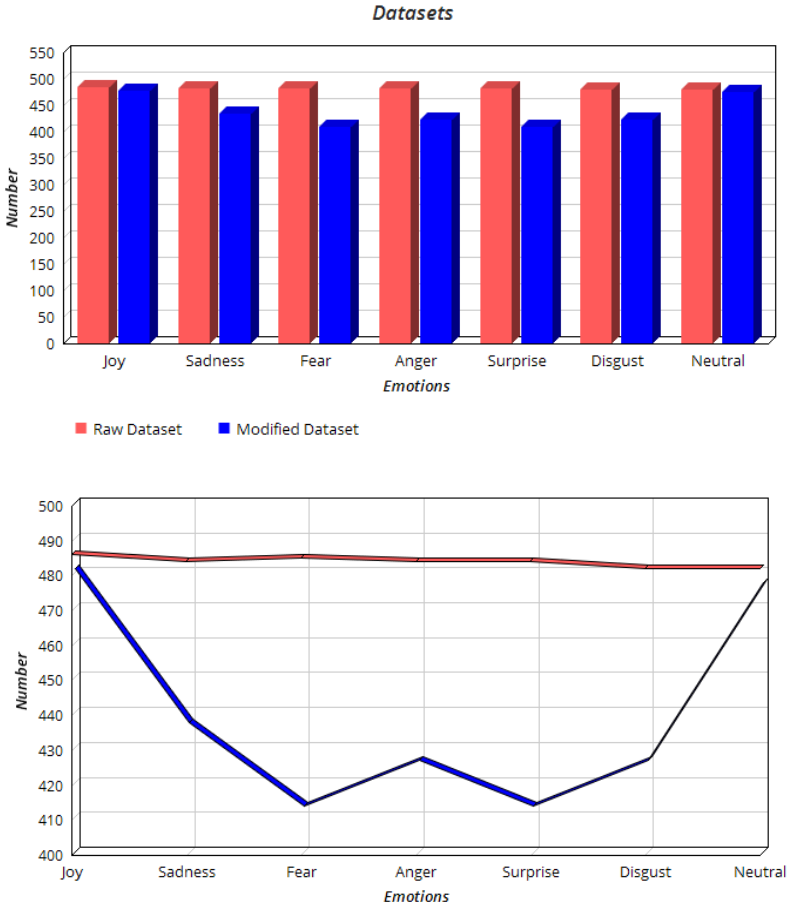

Figure 3.3: *Comparison between Raw and Modified Datasets*

Figure 3.3 shows that in the starting dataset (Raw Dataset) there are just under 500 images per class, for a total of 3394 images, while in the one we modified (Modified Dataset) the number is slightly lower as there are 3090 photos. From this graph it can be seen how the difference between the two datasets is very low regarding *joy* and *neutral*, while it is more evident in the other categories, reaching a difference of over 70 images in *fear* and *surprise*.

Considering all the photos were taken in many different places, with different backgrounds, from different distances and with different angles, which makes training the model much more complex, time consuming and requires much greater computational power, we decided to add one more step as shown in figure 3.2. A normalization algorithm was developed in order to get images more suitable for the training of the CNN.

Figure 3.4: *Process for image normalization*

Image 3.4 shows two examples of how the normalization algorithm works. It can be seen how raw images are processed to reach the final result. The algorithm searches for the faces present in the image, extracts them, recognizes three fundamental points, such as mouth and eyes, and if the faces are not in the frontal position, it rotates them around the three axes to obtain an image as straight as possible. Moreover, the image is converted in gray-scale since colors, according to literature, do not provide particularly useful information for facial emotions recognition. At this point we decided to consider two possible work modalities: the first modality adopts the above described normalization process which provides images like the lower black and white pictures in figure 3.4; the second one adds landmarks to highlight some significant points such as mouth, nose and eyebrows as shown in the upper black and white pictures in the same figure. In the end we obtained two datasets starting from the same images, which gave us the possibility to compare the two and find out if the choice of highlighting critical spots would have improved the learning of the network.

Figure 3.5 resumes all the datasets we obtained: starting from the Raw Dataset, we obtained the modified Dataset where the wrong images were removed. To each of these is applied the two different normalization processes: the first is called "Free Normalization", where the images have been pre-processed in a way to reduce the necessary computational power; the other is called "Landmark Normalization", in which markers have been added to

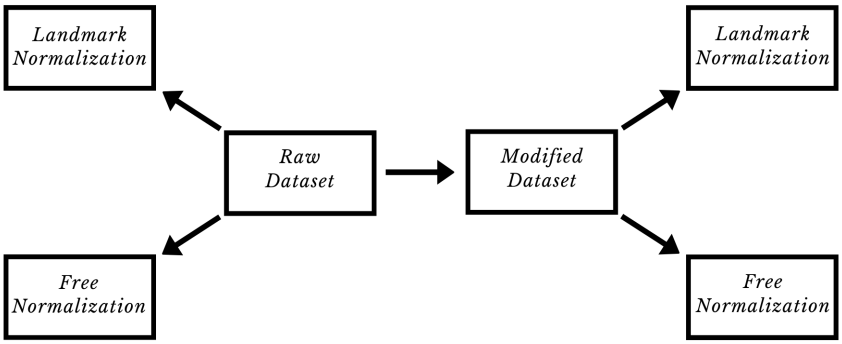

Figure 3.5: *Dataset sub-division*

the previous one to highlight the key points.

Given those datasets, there is still something to emphasize. Among the images we have collected, there are some that are not wrong in terms of emotional expression, but which were unfortunately discarded. In order to properly function, both normalization processes must be able to locate the face and the three fundamental points necessary for the rotation of the latter in a frontal position. Unluckily, in some cases this does not happen and the image is consequently discarded, decreasing the actual number of photographs used for the CNN training.

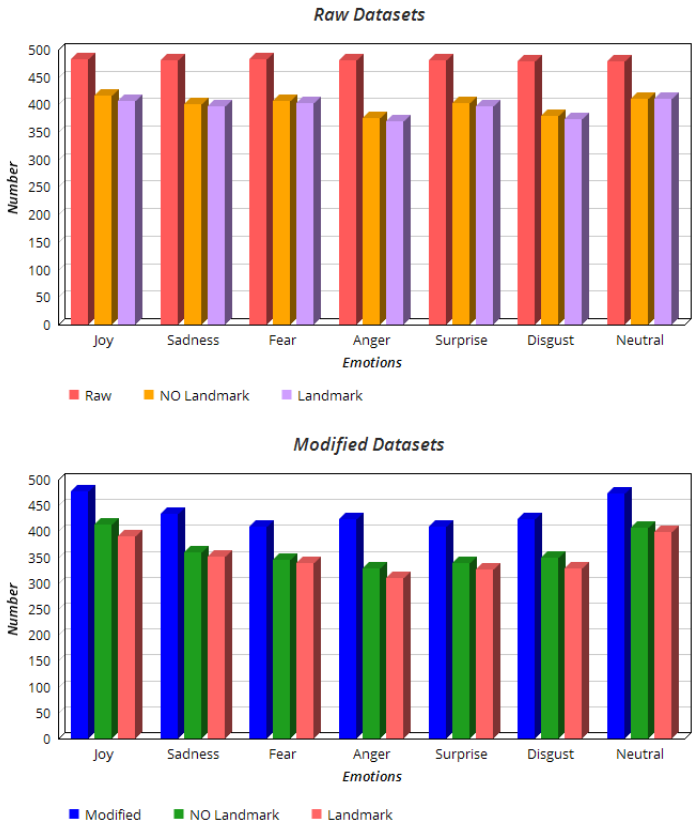

Figure 3.6: *Dataset Comparison*

From the graph in figure 3.6 it is possible to notice how in the landmarks datasets the number of samples is slightly lower than in the one without markers. Even though both were created from the same amount of images, due to the starting position of some faces sometimes it was not possible to find the points necessary for the location of the landmarks, and consequently the image was discarded.

Website

As previously explained, a problem that should not be underestimated when creating a new dataset made by collecting photos of friends, relatives and colleagues, is certainly that of reaching an adequate number of samples. Indeed, as previously exposed, this is a very important factor that could significantly influence the final result and therefore the accuracy of our neural network. To avoid limiting the dataset to a number of samples that was too small, we decided to expand our range by using the countless possibilities that digital technology offers us. We created a simple, intuitive and captivating website[2] where people could be informed about our project and, if interested, contribute by sending us their photos. Finally, to raise the visibility of our work and consequently increase the possibilities of expanding the size of the dataset, we shared the web page on some of the main social platforms: WhatsApp, Facebook, Instagram, Twitter and Linkedin.

Dataset Augmentation

A point we would like to stress a bit more is connected to the number of possibly collectable data. In this case, the choice to create from scratch a collection of images of people makes this point even more delicate since it is an hard and slow job and small datasets often lead to insufficient results. For this reason it could be smart trying to use specific algorithms for dataset augmentation. Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of the images already present in the dataset. Training deep learning neural network models on a wider variety of data can result in more skillful models; the augmentation techniques can therefore create variations of the images that can improve the ability of the models to generalize what they have learned to new images.

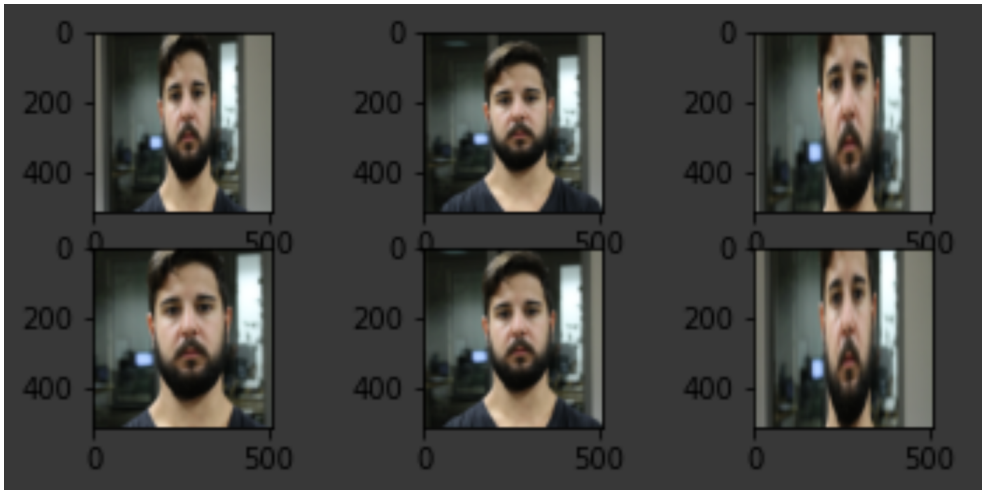

Figure 3.7: *Examples of Dataset Augmentation*

[2] `https://pepemotionrecognition4.webnode.it/`

A big number of Augmentation packages are available and it is of fundamental importance to choose the right algorithm suitable for our aims.

As we can see in image 3.7, the best way to improve the number of images in our dataset is to convert the existing ones modifying brightness, zoom and some face characteristics.

Given that this function called ImageDataGenerator is included in the preprocessing library of Keras [3], we can focus more on the results rather than on the effective steps of the algorithms. We imposed, by modifying some transformation indexes, four different brightness and six different zoom ranges: brightness and zoom amounts are uniformly randomly sampled and, for the zoom case, images are zoomed for each dimension (width and height) separately, in order to obtain, for example, a larger face in case of higher zoom. What we achieve is a consistent improvement of the number of images that is almost six times higher, moving from about 3500 images of "partial-dataset" to almost 19000 of the "augmented partial-dataset". It is important to highlight is that some of the augmented images are lost during the normalization process, because through augmentation transformations some images characteristic are modified so much, like in the case of face deformation, that during the normalization process they are discarded.

This step, however, is very heavy both in terms of computational power and time, therefore it was not possible to apply this algorithm to both the raw dataset and the modified datased. These limitations led us to to create the augmented dataset starting from the one which, as we will see in chapter 4, provided us with the best results. The same thing can be applied to the normalization algorithm: in this case as well we have chosen one based on the obtained data.

Finally, the combination of choice of the datset, augmented process and normalization algorithm made it possible to create a new dataset with an enormously broadened number of samples as shown in the figure 3.8.

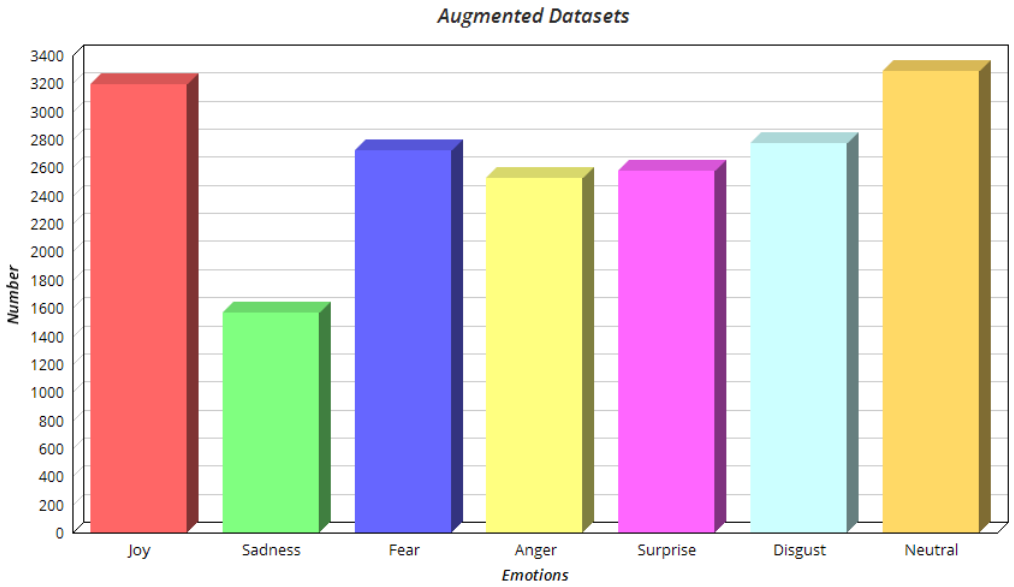

Figure 3.8: *Dataset Augmentation size*

[3] https://keras.io/preprocessing/image/

3.1.2 Convolutional Neural Network

To create the neural network we used a Python library called Tensorflow[4], specifically created to develop machine learning and deep learning models. It is an end-to-end open source platform which has a comprehensive, flexible ecosystem of tools, libraries and community resources that makes it possible to easily build and deploy ML powered applications. To make the task easier we exploited Keras[5], an interface on a higher abstraction level, which is a high-level neural networks API. It is written in Python and capable of running on top of suitable computational libraries such as TensorFlow, CNTK or Theano.

The main advantages of using this multiple level sequence are:

- Possibility of exploiting an easy, fast and adaptable API
- Implementation of deep convolutional networks
- Possibility of running seamlessly on CPU and GPU
- Chance of conversion in Tensorflow Lite to make the most out of TPU

One more very important tool that we decided to use to simplify and speed up the whole process is Google Colab[6], an online platform that allows running codes directly on the Cloud, exploiting the GPU power offered by Google. This platform is very useful, especially in an ML context where, with the increase in the size of the datasets, it makes it possible to execute complex training algorithms otherwise difficult to perform locally. This system is based on Jupyter Notebooks, i.e. interactive documents divided into cells where it is possible to both execute the code and add notes to describe the steps, further simplifying the job, especially in shared project situations. These Notebooks therefore allow, through the creation of a single unique document, both to execute all the steps of an algorithm and to clear up their behavior in natural language.

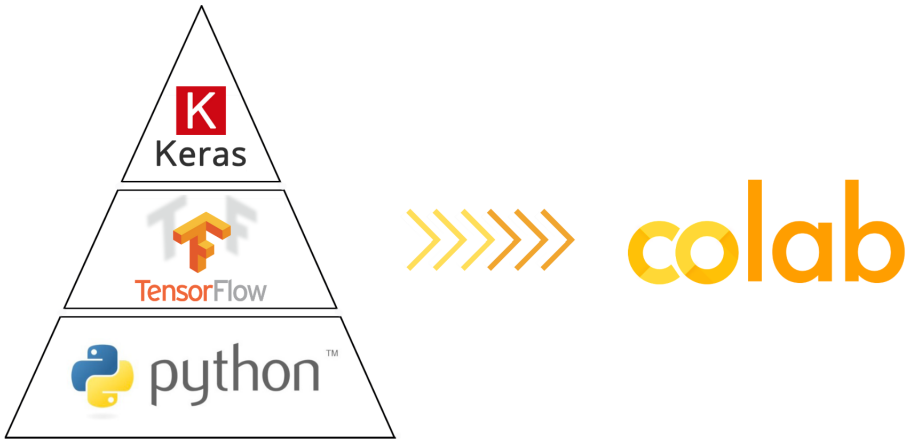

Figure 3.9: *Hierarchical structure for the construction and training of the neural network. Keras built on top of Tensorflow within Python. Running on Colab platform*

By exploiting all the potential provided by the union of Tensorflow and Keras, it was possible to create neural networks suitable for the recognition of facial expressions. To do this, however, we also had to keep in mind the final goal of the project: the creation of a

[4] https://www.tensorflow.org/
[5] https://keras.io/
[6] https://colab.research.google.com/notebooks/intro.ipynb#recent=true

network suitable for running on the GCDB and taking advantage of all the possibilities offered by the Edge TPU, as we will see in the following paragraphs. The decision of exploiting the Edge TPU is not a negligible detail, since it has proved to be a highly limiting factor and has severely influenced the final composition of CNN since there are not many Keras layers supported by this technology. We will then analyze this aspect a little more in detail in the specific section of the GCDB.

Also in this case, as well as for the datasets, we did not focus on a single choice but we decided to analyze different structures to be able to compare the results and choose the one that could provide us with the best results. Starting from a basic scheme consisting of convolutional and fully connected layers, we created four different configuration to be analyzed.

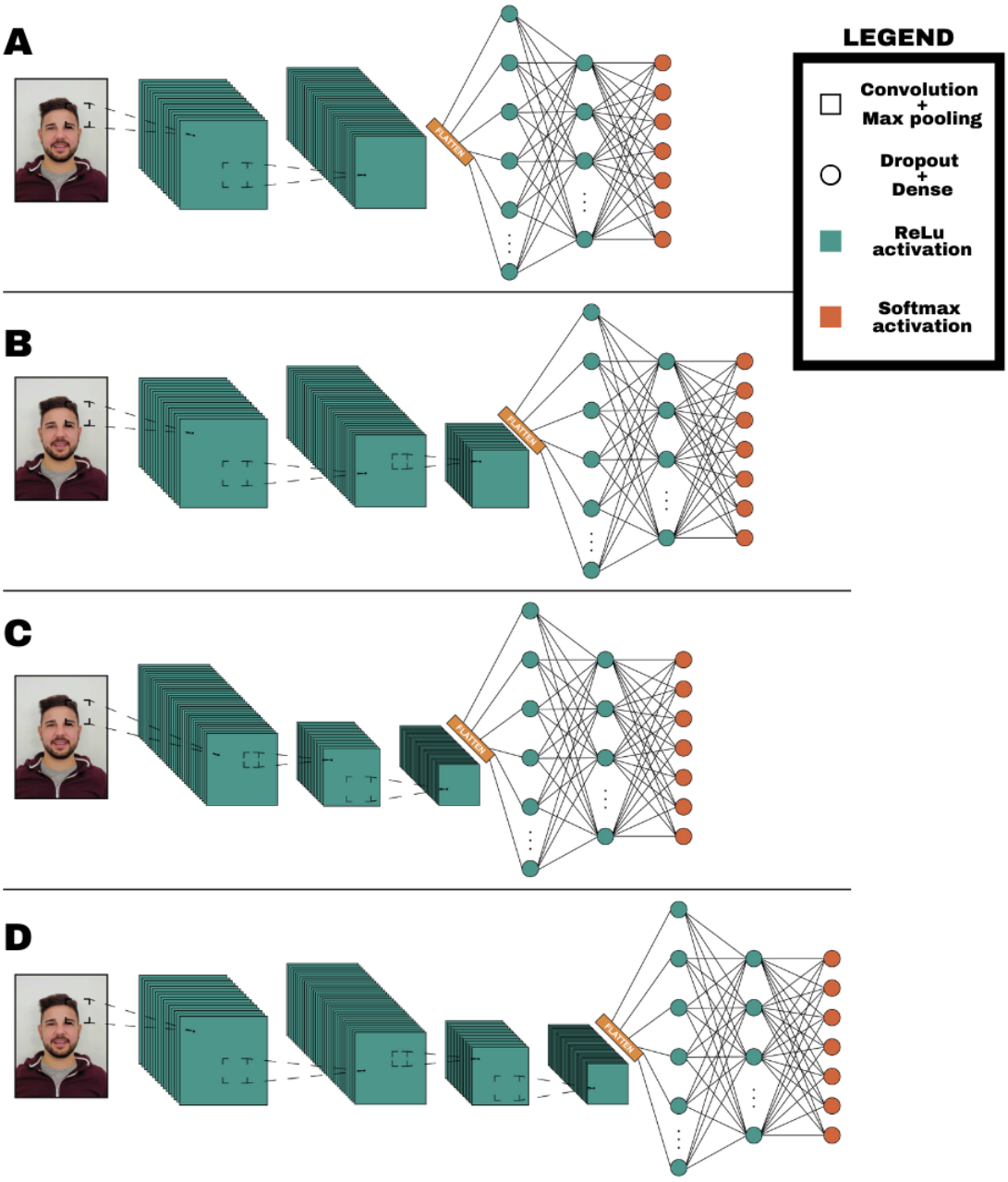

Figure 3.10: *The four different configuration of CNN*

Figure 3.10 shows the four CNN configurations that we have decided to analyze. Each of these is made up of a series of square layers, a rectangular one representing the flatten and three round layers. This graphic representation was created to simplify the explanation of each of them, providing a detailed image for each configuration.

Starting from the description of the last layers depicted as circles, they represent the "dense layers" or rather the "fully connected layers". What can be easily seen, then, is the difference in colors between them. This indicates the different activation methods: for the first two a Relu type activation is used, while for the last one a Softmax type is chosen.

A detail that cannot be overlooked, although not shown in the pictures, is the presence of Dropout layers in front of each dense layer. This is because adding Dropout layers has a positive impact, as they make the network more stable since this technique randomly selected neurons to ignore during training.

Starting from the top we can name, as seen in figure 3.10, the four configurations as "A", "B", "C", "D". We can see that the only aspect that makes each of these networks different is the number of convolutional layers, represented as squares, and their configuration. In this case the use of colors was not random, as the chromatic effect suggests an equality in the choice of the type of activation: Relu activation type was chosen for each layer regardless of the configuration adopted.

As it can be seen, the "fully connected" layers repeat themselves unchanged for all four structures, so we can analyze them a little more in detail, focusing only on the first part: the convolutional layers.

Configuration A consists of only two convolutional layers: the first including 16 filters with a 5x5 convolutional window, while the second has an exactly doubled number of filters with a reduced convolutional window showing dimensions of 3x3.

Configuration D, on the other hand, is nothing more than a double repetition of the two layers just described. In fact, it shows an alternation of convolutional layers with 16 filters and layers with 32.

For what concerns configurations B and C, they are both made up of 3 convolutional layers. In case B the last of the four layers of D is removed, while to obtain case C the first one is eliminated. This way we obtain two 32 filter layers interspersed with a 16 filter layer for C and two 16 filter layers divided by a 32 filters layer to form B. Not shown graphically, each convolutional layer is associated with a pooling layer.

Now that all parts are available, it is possible to start with the training.

For this phase, each of the aforementioned datasets was used to train each of the CNN configurations just described, so as to be able to compare the data obtained and find the combination that provided the most satisfactory results.

TF lite Model

Once the training phase is over and the best neural network configuration is found, associated with the dataset providing the best results, we need to convert the model. As already briefly suggested, in order to work on mobile and embedded devices CNN must be converted to TF lite, a version of the principal platform designed to guarantee greater lightness and speed compared to the main release.

In this specific case, the simple conversion of the model into Tensorflow lite format is not enough to exploit the possibilities offered by the Edge TPU, as can be seen from the table in figure 3.11 taken from the official website[7]. In order to be executed on the TPU, the model must be optimized first by following a " Full Integer Quantization ", which makes the model

[7] `https://www.tensorflow.org/lite`

Technique	Benefits	Hardware
Dynamic range quantization	4x smaller, 2-3x speedup, accuracy	CPU
Full integer quantization	4x smaller, 3x+ speedup	CPU, Edge TPU, etc.
Float16 quantization	2x smaller, potential GPU acceleration	CPU/GPU

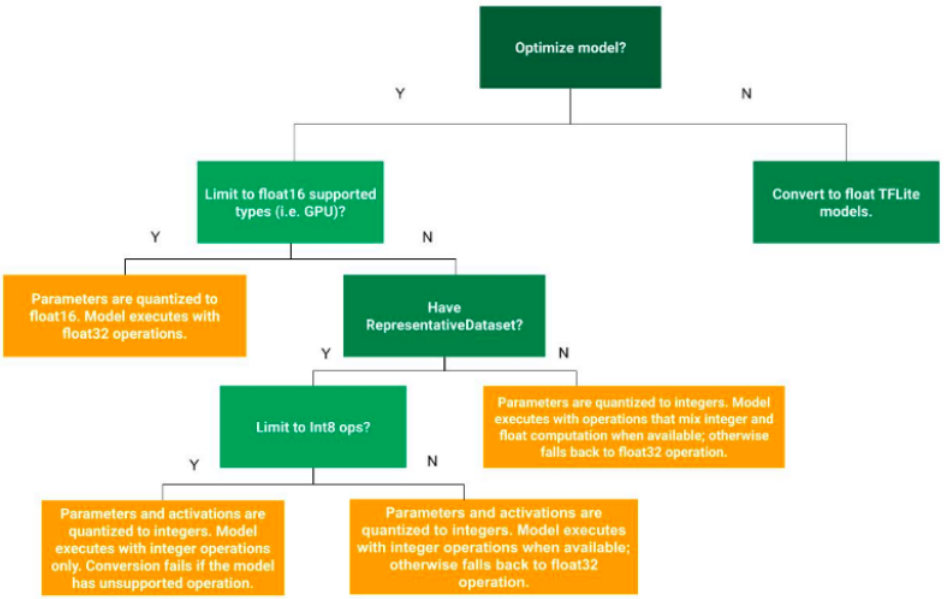

Figure 3.11: *Decision tree to determine post-training quantization method*

four times smaller in terms of computing power required, and makes it up to three times faster. These reported optimization methods are post-training quantization algorithms, i.e. the optimization of the model is applied only after the end of the training phase.

As we can see from the decision tree, to get a fully quantized model it is necessary that all parameters and activations are quantized to integers and the model executes with integer operations only. If any unsupported operation is present into the model the conversion fails. So, once the optimization passages were completed, we converted the network to obtain a model in tflite format, suitable for running on our embedded device, using all the computational power offered by the TPU processor.

3.2 Google Coral Dev Board

Starting from figure 3.1, it is possible to see that the second step involves the implementation of CNN within the Google Coral Dev Board [8], a device whose hardware has been developed expressly to speed up the use of neural networks. A Tensor Processing Unit (TPU) is present within the device, which is an AI accelerator consisting of a circuit ASIC developed by Google specifically for neural network machine learning, particularly using Google's TensorFlow software. This chip was designed as a matrix-processor exclusively to work for neural networks at incredibly high speeds while consuming much less energy and within an incredibly small physical space. In addition, the device is equipped with a standard CPU processor which

[8] `https://coral.ai/products/dev-board/`

makes it able to perform preprocessing steps to handle the video stream and to extract suitable images; these images will then be elaborated in order to provide the appropriate output for the robot management.

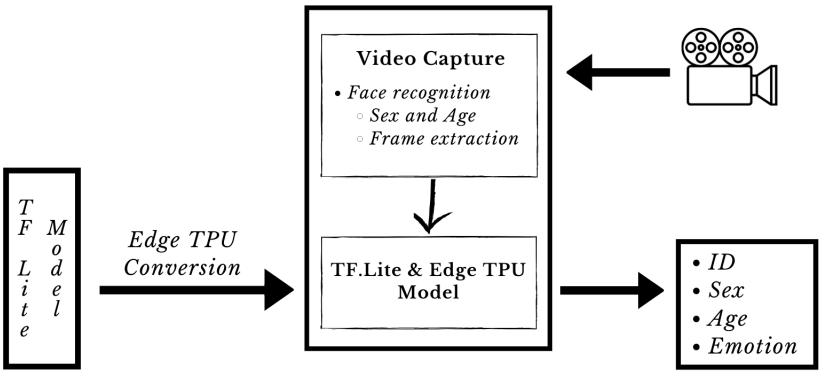

Figure 3.12: *Google Coral Dev Board logical flux*

GCDB is the core of our project, since right on this device has been loaded the source code, which is the main code that links the acquisition from the video source, the analysis of face features and the emotion recognition.

Indeed, figure 3.12 shows the work that GCDB does to make our project work. Video capture, a program for managing the video stream has been developed. It analyzes the incoming video and if it recognizes the presence of faces in the frame, deduces subject's gender and age, extracts the frame of interest and extrapolates the faces. Once this extraction process has been carried out, each single face is examined by an inference created specifically, using the model described in the previous section, to derive the emotion. At this point all the obtained data, are combined within a class to form a new type of data used then for robot interaction.

3.2.1 Video Capture

This is the part of the code which is involved in the management of the video. All the code is optimized to use Python libraries and to allow the interpretation of the converted NN model. The code is designated to the acquisition of the video frames and their preprocessing, so to allow the correct and real-time FER. It is structured in multiple functions, each of them involved in the envelope of the main features of the project:

- Video acquisition
- People recognition
- Frame analysis and transformation
- Face encoding
- Age and gender detection

OpenCV[9]

OpenCV (Open Source Computer Vision) is a free, software and cross-platform library for the artificial vision in real time, that is the whole of processes to create an approximation

[9] `https://opencv.org/about/`

of the real world (3D), from bi-dimensional images (2D). The programming language mainly used with this kind of library is the C++, but it's possible to interact with it through Java and Python languages, as in our case.

OpenCV 2.4 is the actual version containing the new class for face recognition, which contains all the algorithms for recognizing and distinguishing between known faces. The approach is geometrically, indeed marker points (i.e positions of eyes, nose, ears) were used to build a feature vector (i.e distances or angles between the points). For each function of the library, a specific transformation is applied to the vector.

Face Recognition[10]

As we have seen above, OpenCV library allows us to acquire a real time video from an external source, using the communication protocol (HTTP - Hyper Text Transfer Protocol), which lets client and server transfer data to one another. In this way the device and the camera can communicate through a specific HTTP, creating a real time video stream.

The video streaming is controlled frame by frame as long as there are no faces in the image, while, whenever a person is found, the face is localized and cropped, keeping in memory the face features trough OpenCV "face-recognition" algorithm.

Now we need to make a distinction between two categories of subjects: those who are framed, but who are only passing through, and those who actually stop to interact with the robot or to observe it for a longer time. In order to make this distinction we inserted a timer. When a person is detected, the face is localized and the program checks if the permanence is temporary or not before starting with the actual image processing. Therefore to understand if a subject can be considered "interested", a three-second count (arbitrarily chosen value) is activated from the moment the face is detected. Once the threshold value is reached, if the face is still present in the frame, the subject is considered interested and the image processing and analysis can be activated.

The first step, once the subject is marked as interested, is to elaborate the image to make it suitable for all required passages, included the FER through the CNN. For this reason a gray-scale conversion and a change in resolution are needed. This way every single face is converted in black and white and is reshaped to match the same size of the images used to train the network.

Figure 3.13: *Real time face extraction*

[10] `https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html`, Face Recognition with OpenCV

Figure 3.13 shows how the image is processed when a subject is localized into the frame: colors are removed, the original image is cropped into the face only and new sizes are set.

Although the project is primarily designed for the analysis of a single subject, it cannot be excluded that more people could be included into the same camera frame. For this reason we decided to also focus on the management of multiple faces taken from the same image.

We created a function able to recognize and analyze more than one person simultaneously, so that it is possible to correctly link all the data to each person recognised. This step is of fundamental importance, since it allows us to correctly point out the emotions shown on different faces. At first, people are recognised through the encoded characteristic of the face (distance between nose and mouth, distance between eyes and ears, width of the mouth, etc.), then they are memorized by the "face-encoding" algorithm, present in the Face Recognition library. Lastly, an ID is linked to each person.

By associating each person with his or her specific ID, it is possible to obtain advantages on various aspects, from dealing with the simultaneous presence of multiple faces in the same image to the final management of all data collected. In addition to uniquely associating each person with their own characteristics, this function also allows to memorize people already seen during the whole session.

If enabled, a specific function is able to compare the incoming faces to those already stored. This way, if a match is found it means that the subject was already analyzed; if this is the case, the subject gets associated to their previous ID, otherwise a new ID is assigned and, with it, the new face is memorized.

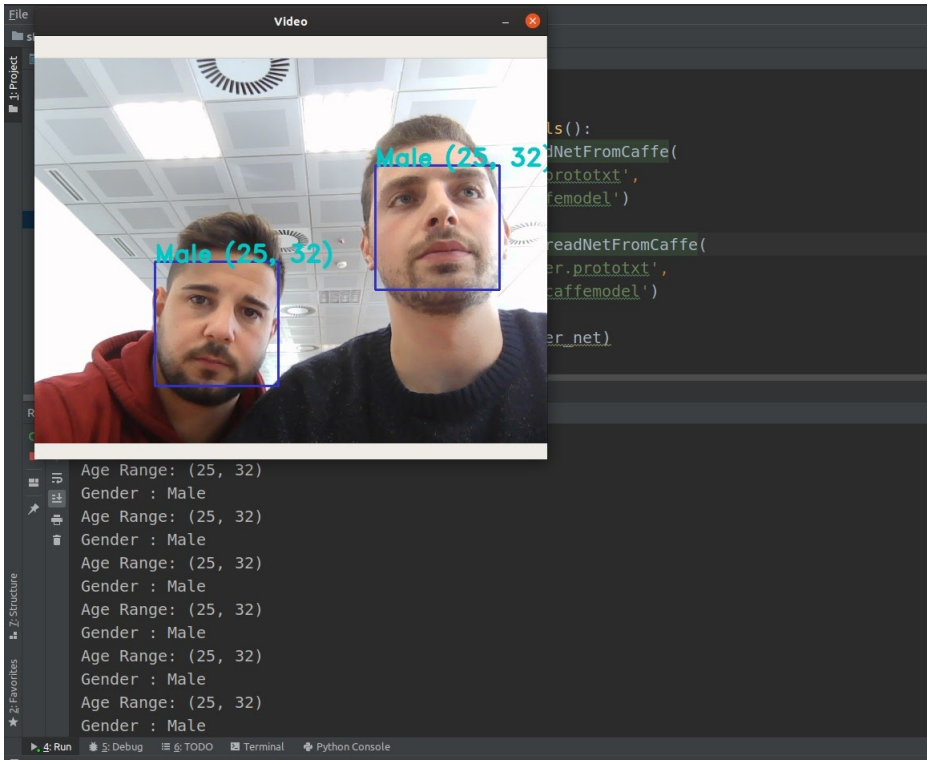

Figure 3.14: *Age and Gender Detection*

Once the image is processed, the faces extracted and saved and the ID assigned, the next step is the use of two specific algorithms for age and gender detection. These classifications are not strictly correlated with FER but are of major importance when it comes to the final goal of increasing robot empathy and profiling subjects. The predicted gender obviously may

be one between "Male" or "Female" while the predicted age may belong to the following ranges: (15-20), (22-32), (33-43), (44-53), (60-100). The way to accurately predict age and gender is the use of pre-trained "cafe_model" algorithms, existing in the face_recognition libraries.

The difficulty to accurately guess an exact age from a single image, due to influencing factors like make-up, lighting, obstructions and facial expressions, led us to make the choice to mark it as a classification problem instead of a regression one, classifying the ages in more ranges and not in a single age, making the computation more reliable.

Finally, after the preprocessing of the frame and the analysis of the useful data, the image is ready to be processed by CNN to extract the most relevant data: the emotional state.

3.2.2 Edge TPU Model

In the previous paragraph regarding CNN we explained how the optimization and conversion of the model made our model suitable for execution on embedded devices equipped with TPU. Although that passage was fundamental, it was not sufficient for the correct release of the model on the TPU, therefore a further step is required. At this point, the already trained, optimized and converted model must be compiled in order to make it suitable to exploit the entire computing power of the device. An Edge TPU compiler is requested to preform this task.

Figure 3.15 shows a file, generated by the compiler, that presents the respective layers actually mapped to the Edge TPU. It can be seen that two of these have not been effectively mapped like all the others and that they were not physically inserted into the structure of the neural network: "quantize" and "dequantize". These two layers are automatically added by the compiler as the conversion process has not yet been optimized to support Tensorflow 2.0 used for the creation of the entire system, but according to what specified online, their presence should not modify the device's performances.

```
Edge TPU Compiler version 2.0.267685300
Input: try.tflite
Output: try_edgetpu.tflite

Operator                       Count      Status

MAX_POOL_2D                    4          Mapped to Edge TPU
QUANTIZE                       1          Operation is otherwise supported, but not mapped due to some unspecified limitation
CONV_2D                        3          Mapped to Edge TPU
DEPTHWISE_CONV_2D              1          Mapped to Edge TPU
DEQUANTIZE                     1          Operation is working on an unsupported data type
SOFTMAX                        1          Mapped to Edge TPU
FULLY_CONNECTED                3          Mapped to Edge TPU
```

Figure 3.15: *File showing layers mapped to Edge TPU*

The compilation of the model is a particularly complex task; there were many attempts that we had to make before being able to obtain models that were completely compiled on the TPU. There are still many layers and combinations of layers that are not supported on this new technology, which made the creation of a fully compatible network quite complex. As previously briefly described, GCDB is also equipped with a CPU able to perform all the tasks that cannot be performed at the TPU level. Hence, when the compilation is not totally matched, the steps that are not mapped to the TPU are executed by the CPU. This slows down the whole process, thus nullifying the improvements made by the introduction of the new technology.

Once the model is completely mapped to the TPU, it is possible to exploit the entire computing power available on the device using a TF Lite function capable of creating inferences. This function is called TF Lite Interpreter. Its purpose is that of recalling the model which

is then able to create an interference that processes the input images into the desired data, in this case the emotion of the subject.

TensorFlow Lite Inference is, indeed, the way to correctly run the TF Lite converted model. The term inference refers to the process of executing a TensorFlow Lite model on-device in order to make predictions based on the input data.

TF Lite Interpreter is designed to be lean and fast, using a static graph ordering and a custom (less-dynamic) memory allocator to ensure minimal load, initialization, and execution latency. Moreover, the use of Interpreter takes advantage of the possibility to not install all the TensorFlow and TF Lite libraries on the device, reducing the usage of memory.

```
# Load TFLite model and allocate tensors.
interpreter = tflite.Interpreter(model_path = "edgetpu.tflite",
                                 experimental_delegates=[tflite.load_delegate('libedgetpu.so.1')])
interpreter.allocate_tensors()
# Get input and output tensors.
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
age_net, gender_net = initialize_caffe_models()
print('\nInizialization ok')
```

Figure 3.16: *TFLite Interpreter extract from code*

Referencing to image 3.16, it can be seen that only few steps exists to initialize and run the model on the Interpreter:

1. Loading a model (.flite version) into memory
2. Transforming data to resize an image and change the image format to be compatible with the model
3. Running inference using the TensorFlow Lite API to execute the model: it involves a few steps such as building the interpreter and allocating tensor
4. Interpreting output tensors in a meaningful way that is useful in the application needed.

3.2.3 Emotion Recognition

With all the useful elements now available, it is possible to perform the most important step: the emotion recognition.

The Interpreter creates the appropriate inference and the image can be analyzed. At this point, however, it is also necessary to take into consideration the methods that lead to the effective analysis of the data. In fact, it was necessary to create a protocol in order to be able to manage the work in an optimal way. The factors that played a key role in the development of this protocol the most are: the time, the simultaneous management of multiple faces, which we already discussed in the previous paragraph, and the computational power made available by the device.

At the beginning we decided to elaborate and to analyse every single frame where a person was detected, but considering the transmission of the video is about 25 fps, we soon discovered that the computational power required for elaborating the full pipeline was too heavy. Consequently, it is not possible to perform all the preprocessing steps (people detection, face recognition, gray scale conversion and reshaping, face extraction, face memorization, age and gender detection) combined with FER for every single video frame.

Giving this limitation, we considered that the use of all the 25 fps is not really necessary, since even without considering every single frame the general perception of the subject feeling is not compromised. In fact, we are interested in the long term feelings and even more in recognising changes in the emotional state that a person can express in a real life situation.

For this reason we decided to use a different approach and to set an arbitrary waiting time interval. When a person is detected and marked as interested the algorithm starts, all the steps are executed and all the necessary data, emotion included, are extrapolated. Now a counter is set and every 2 seconds a new analysis is executed until the face is present in the frame. We decided to adopt and arbitrary time range of two seconds, since it is neither excessively wide (this avoids the risk of losing too many important details) nor too short (thus avoiding problems in terms of device performance). Keeping in mind that ignoring part of the frames could cause information loss, we considered this possibility as a negligible error. The time interval can be nonetheless modified to adapt the process to any eventuality.

Before moving on, it is important to consider one more scenario: the one in which a new person in localized during the "waiting interval". As this is a far from trivial issue, two ideas were investigated. The first one proceeded with waiting until the counter reached the threshold before analyzing all the faces on the screen; in the second one the counter would stop in order to analyze the faces available on screen in that moment and then reset itself. Eventually the choice fell on the second option: we preferred giving priority to the arrival of new people in order to minimize the error due to the loss of frames.

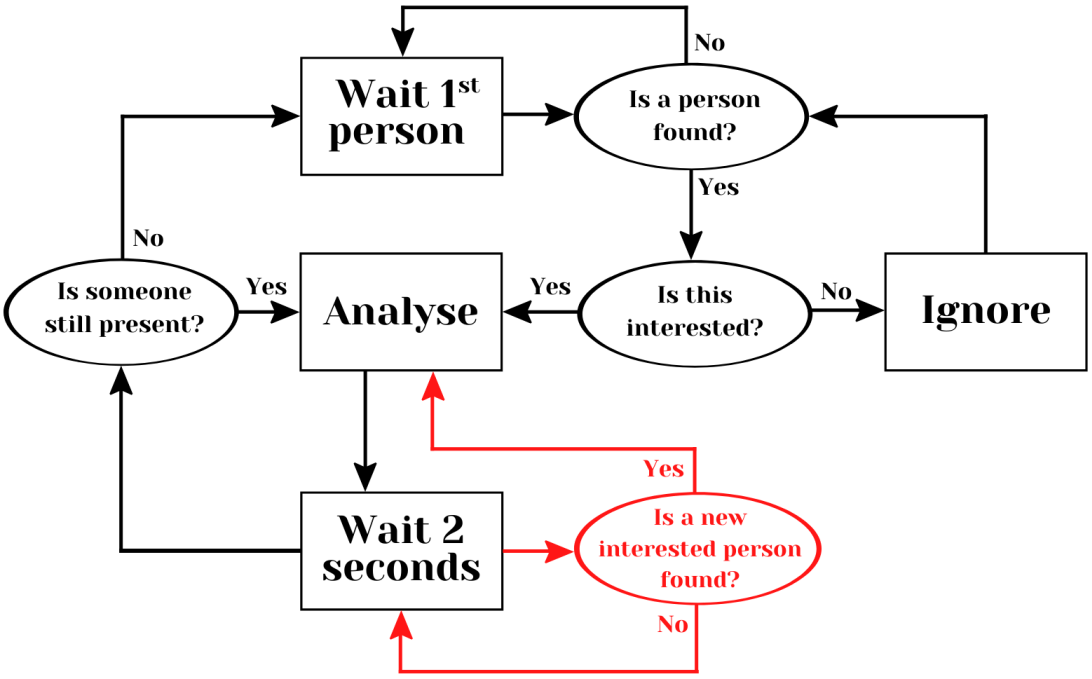

Figure 3.17: *Block diagram explaining how the algorithm works*

Therefore as can be seen from figure 3.17, once a person is localized and identified as interested, the image processing starts, leading to obtain all the useful data. Once this first analysis is complete, a waiting period of 2 seconds begins, after which if the person is still present a new analysis is carried out. This repetition of the waiting and the analysis phase can only be interrupted when the subject leaves the field of view of the video camera. As can be seen from the image, during the waiting period, an asynchronous event can occur. It is indicated in red and it is independent of the count of the time that regulates this phase.

This event is the arrival of a new person within the frame. In this way the waiting phase is immediately interrupted and the cycle restarts from the analysis.

Each time the analysis phase is completed, the data obtained for each person within the frame are manifold, so for convenient management we decided to enclose them within a single element. We created a new python class called "Person" where the obtained data could be divided and organized.

This object Python class is structured as follow:

1. ID
2. Age Range
3. Gender
4. Emotion

The four elements that make up our data object provide all the useful data that allow us both to increase the empathy of the robot and, thanks to the use of statistical analysis, to profile the various interlocutors.

The ID is really important, because thanks to this we are able to uniquely identify each individual subject that comes in contact with the robot, thus avoiding an incorrect data assignment. Each face is recognized and associated with the respective parameters of gender, age and emotion. The last three parameters, unlike the ID that is associated only at first, are updated whenever a face analysis is performed. This allows us on the one hand to know the changes in the emotional state of the subject over the time, on the other hand make it possible to compensate and exclude possible incorrect values.

In conclusion whenever the person analysis is completed and the object class is created or updated, a post-request to the HTTP of the server is done, and all the objects are send in JSON format file.

Summarizing, for each person capture by the camera :

- The ID of the person is memorized with the face features encoding
- Age and Gender are evaluated
- Emotion are computed by the CNN
- An object class is created to simplify data management and sending to the database

3.3 Robot Interaction

This is the last phase shown in figure 3.1. This section explains the communication between the external device and the robot, a fundamental step to manage Pepper's communication skills, increasing his empathy.

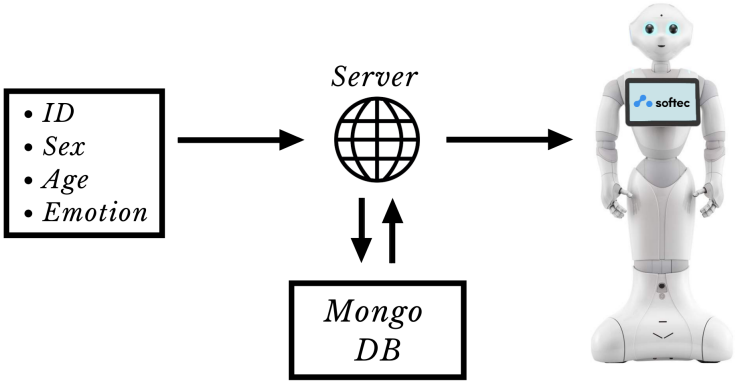

Figure 3.18: *Pepper Process*

Figure 3.18 shows how the data generated at the GCDB level is managed and shared through a server. This information therefore can be saved in a database both to profile the various subjects thanks to elaborations carried out at a later time and to menage the robot during the conversation with people. To improve this ability Pepper have to know the data relative to its interlocutor in order to create a more specific, subjective and personal connection with the human, greatly increasing his empathetic ability.

3.3.1 Communication protocols

In order to create a connection with the GCDB we exploite Node.js, a JavaScript runtime guided by asynchronous time events. It is developed for creating scalable network applications.

The object class, created by the source code and containing all the data we are interested in, is converted in a JSON object. This converted object is then sent to the database thorough an expressly created local server. The server, running in Node.js, is nothing more than an HTTP instance that works locally and when receives a "request" with the argument of the source code, the JSON object in our case, sends a response. The JSON class is finally sent to MongoDB, a database where every data is saved as in figure 3.19.

Simultaneously, the server is waiting for a "send request" from Pepper: when this request arrives, through a specific query the server extracts the emotion parameter from the last data sent to MongoDB and sends it to the robot.

3.3.2 MongoDB

Mongo is the most popular database for modern applications, it is general purpose, document-base, built for modern applications and cloud storage. It is classified as type NoSQL because of its dynamic scheme using JSON documents, differently from the traditional Relational Databases.

Figure 3.19 shows how the data collection is structured: all the data are stored in the same root, classified per date and time.

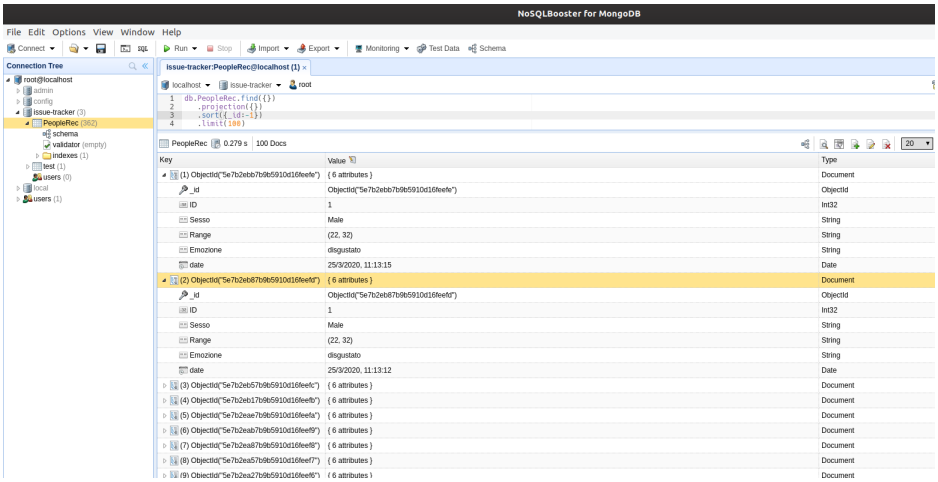

Figure 3.19: *NoSQLBooster MongoDB*

The management of the data is very simple, with a multiple choice of functions that can be called in a query to perform multiple tasks. The database is structured to easily analyse, organize and do statistics on data. It is possible to regulate the entire amount of data, to extract all the same ID emotions, to evaluate the changes in the humor of a person, to analyse all the emotions in a specific range of ages or to divide male from female, older from younger. In this way it make easier to find out the statistical nature of the data.

3.3.3 Pepper

Pepper is one of the most advanced robot system planned for human interactions. It is able to understand humans, to have conversations and entertain. It is an innovative connection between AI and humanoid robotics which exploits the easiest communications forms, such as voice and gesture.

Pepper is developed and produced by SoftBank Robotics and, then, strengthen in applications and behaviours by Softec S.p.a.

The BMS (Behaviour Management System), a module of the Orchestra IoT Platform, is the system that allows to make Pepper doing some advanced tasks.

Orchestra is a cloud based platform for collecting and correlating physical and digital data in order to deliver information, anonymous or highly profiled, useful to improve the customer experience. It sensibly reduces the complexity in designing, developing and managing humanoid robots such as Pepper, by enabling the creation of behaviour that the robot executes, based on users' inputs or programmatic events.

Orchestra Robotics guarantees the programming, installation and robots' bunch management, allowing analytic data collection individually picked up from each robot.

The BMS as well is an advanced RSO (Robotic Service Orchestration) tool, based on the following points:

1. Cloud based platform: no coding and programming skills required to create an application

2. Modular and highly customisable

3. Allows to install, update and execute applications anywhere, on single or multiple robots

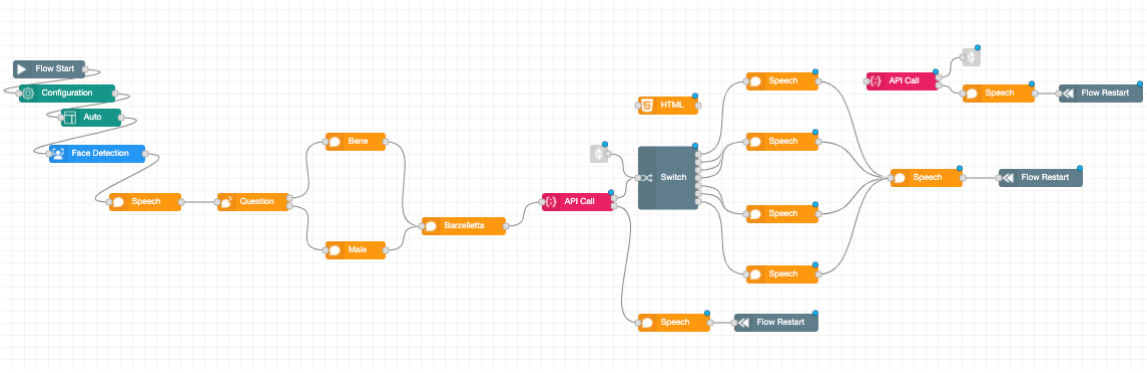

Figure 3.20: *BMS modules flowchart*

The BMS Designer is made up of several modules and the flux of behaviours is structured as a flowchart. Each module is a behaviour and linking them together allows the user to create a completely customised customer experience that provides full interaction:

- voice conversation
- visual information
- gestures
- tactile
- proximity activation
- 3rd party platform/service integration

From the administrative interface, the applications are automatically synchronised on the robot, that executes them via the BMS Executor. During the execution, data, behaviour and logs are collected and displayed in real-time on Orchestra Analytic Dashboard.

In Figure 3.20 an example of execution flowchart is shown: the most significant parts for this thesis project are the API Call modules, where the system makes a request to the server, through HTTP address, and receives the last emotion stored in the database. The emotional information, received as an integer, is translated in emotion type and in this way the switch module directs the flux of the flowchart to a specific action. Following this scheme the robot. Following this schematic, the robot is no longer limited to always carrying out the same commands, but its actions may vary according to the emotional state of the interlocutor. In this way it is possible to manage Pepper's reactions in accordance with the emotional state of the subject, making it a highly empathetic robot.

Final Scheme

With the last image shown in figure 3.21, we want to present a general summary of the entire path carried out during this thesis work. The three steps reported at the beginning of each subchapter have been united to form a graphic representation capable of showing all the steps in a single scheme, making it easier to understand the general picture.

Starting from the top, all the steps necessary for training the model and for its first conversion to TF lite are found. Following the scheme, you reach the GCDB which turns out to be the actual "engine" of the whole system as both the model, further converted to

adapt to the TPU, and all the algorithms necessary for the management and analysis of the video streaming are loaded. This is able to create a wireless connection to a video camera, receive the transmitted frames, process them and, as output, provide data containing all the information useful to achieve the final goal.

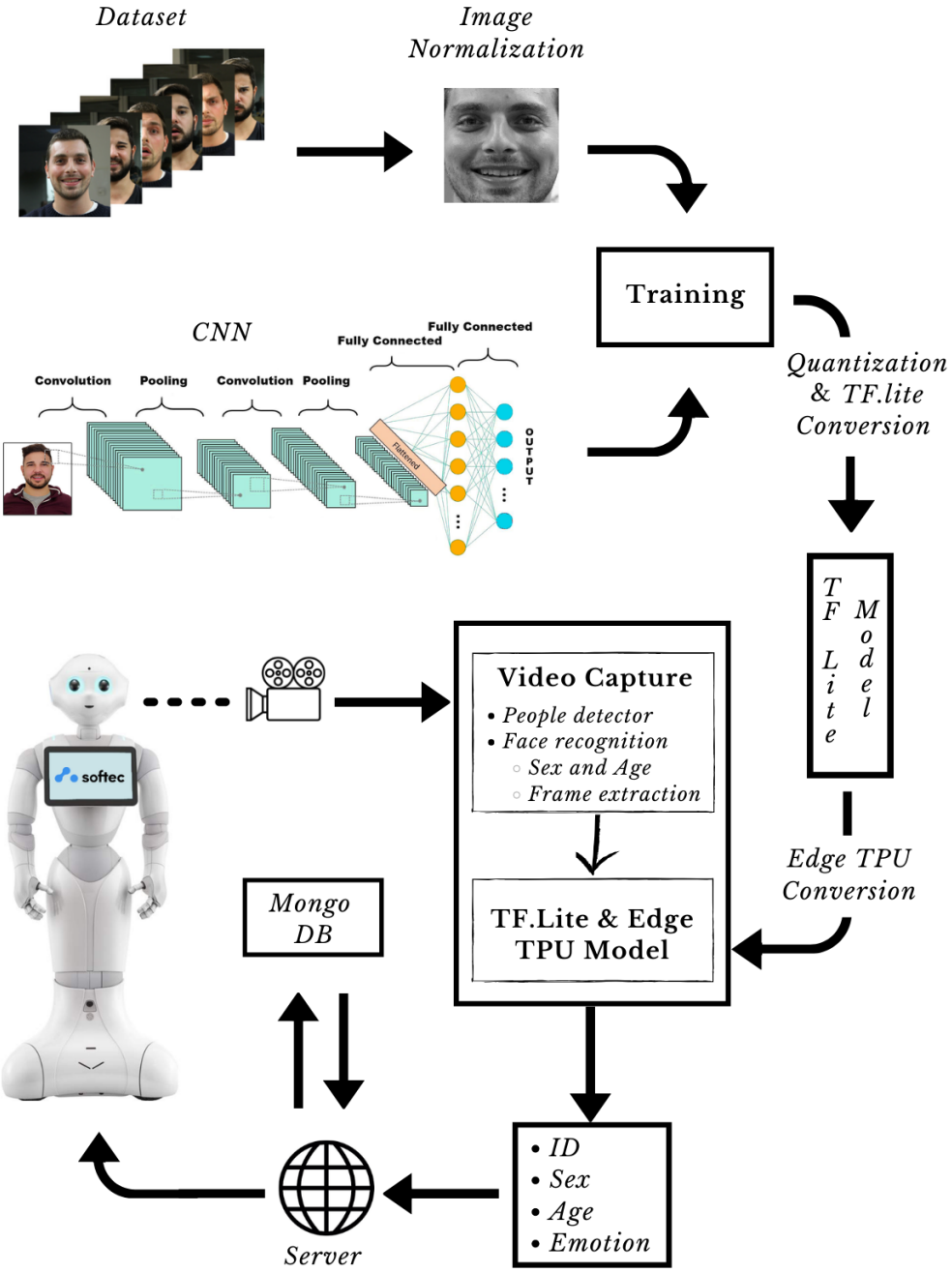

Figure 3.21: *Final schematic representing the whole process*

The output of the embedded device is then sent to a server, which holds the task of managing both functions for which the data was created. On the one hand it communicates with the robot to guarantee access to data whenever required, on the other it is connected to a database, where each value is saved and stored, ready to be subjected to a more in-depth analysis for profiling of all the subjects who came into contact with Pepper.

The last point to underline is the dotted line that connects Pepper to the video camera, thus closing the ring. This link indicates that although our system can be adapted and connected to any video camera, for this project we took advantage of the images from the integrated camera on the robot.
Pepper in this way turns out to be the first actor of the process since it is precisely from the robot camera that the images are obtained, but also the end-effector as the data provided are used to govern his communication skills and increase its empathetic abilities.

Chapter 4

Results

4.1 CNN Performances Analysis

4.1.1 Raw Landmark Dataset

This section lists the results obtained following the training phase using the "Raw dataset" created through landmark normalization. Just to make a brief reference, the raw dataset is the collection of all the images without any modification or correction.

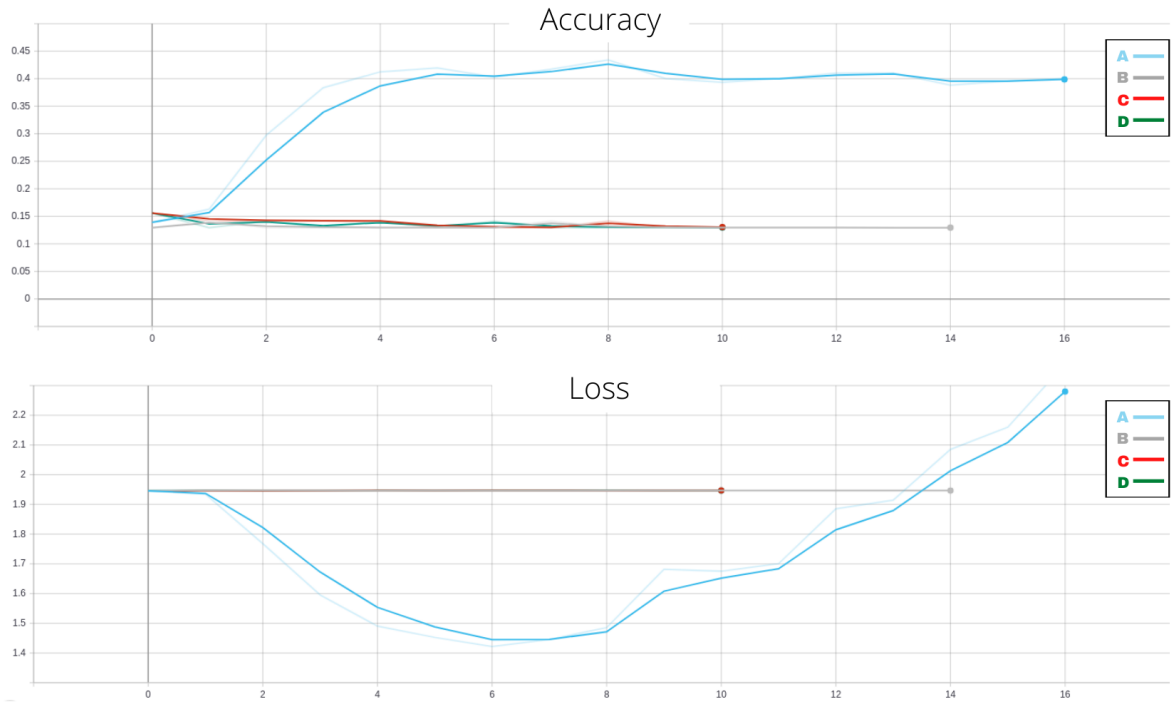

Figure 4.1: *Charts representing accuracy & loss of landmark Raw Dataset for each CNN configuration*

Figure 4.1 shows the accuracy and loss graphs obtained by training each of the four neural network configurations. These curves represent the validation values for each configuration and are obtained by splitting the starting dataset into two parts: 85% used for the actual training, while the remaining 15% to verify the effective performance of the system.

Looking at the graph obtained using this type of dataset, combined with configuration A, we can actually notice a curve indicating an improvement in the accuracy of the predictions, which is now able to correctly identify about 40% of the cases analyzed. Associated with

this, however, the loss graph should also be observed, which shows, after a short descending phase, a new increase until it even exceeds the initial values.

On the contrary, when we look at both the accuracy and loss data in the other three cases, roughly stationary lines are highlighted which indicate that the model was not able to improve its performance during the training phase.

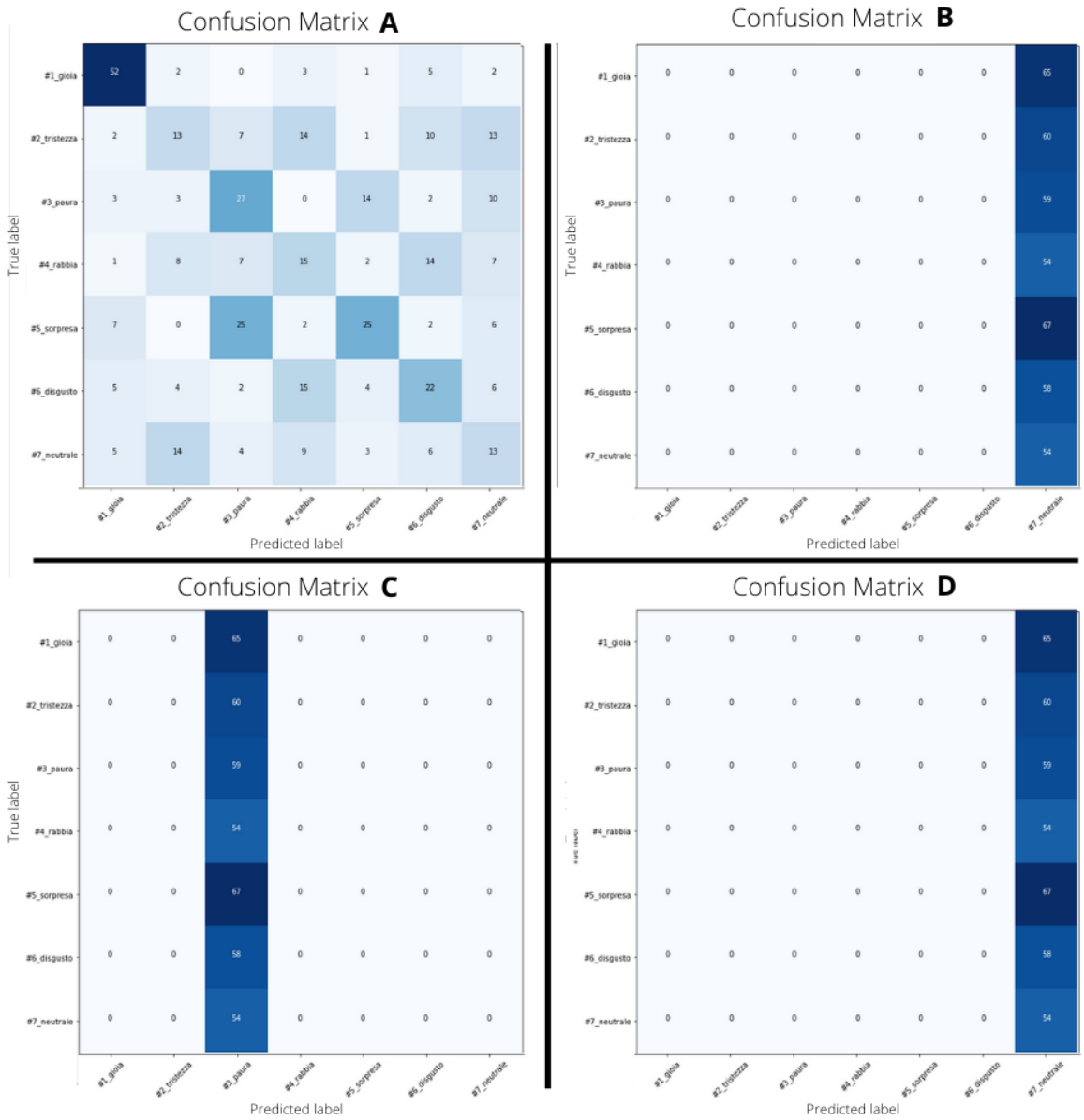

Figure 4.2: *Confusion matrices of landmark Raw Dataset for each CNN configuration*

The squares represented in figure 4.2 show the confusion matrices resulting at the end of each training phase. Starting from the top left, it is possible to notice in each configuration how precise the prediction was compared to the real emotions.

The presence of these confusion matrices is very important as considering how they are able to provide values relating to each individual class, therefore allowing us to understand if some emotions are more easily recognizable than others or among which ones the risk of error is larger.

Complying with what is seen in chart 4.1, the CMs in tables B, C and D show how the

predicted labels belong only to a single category, indicating an almost completely random assignment. Configuration A is the only case that gives us useful information to analyze. In this case it can be seen that joy is almost totally indicated correctly with 52 correct and 13 incorrect. On the contrary, sadness is the most complex case since it is indicated almost equally as either sadness (13), anger (14), disgust (10) or neutral (13). Furthermore, surprise is often mistaken for fear (25 are cases indicated as the first and 25 are erroneously indicated as the second).

4.1.2 Raw Free Dataset

This section instead, is made, once again, by the training through a Raw Dataset, but this time the free normalization algorithm was applied. Always making a brief recall, with free normalization we mean the process that makes the image suitable for training, without however highlighting its fundamental points.

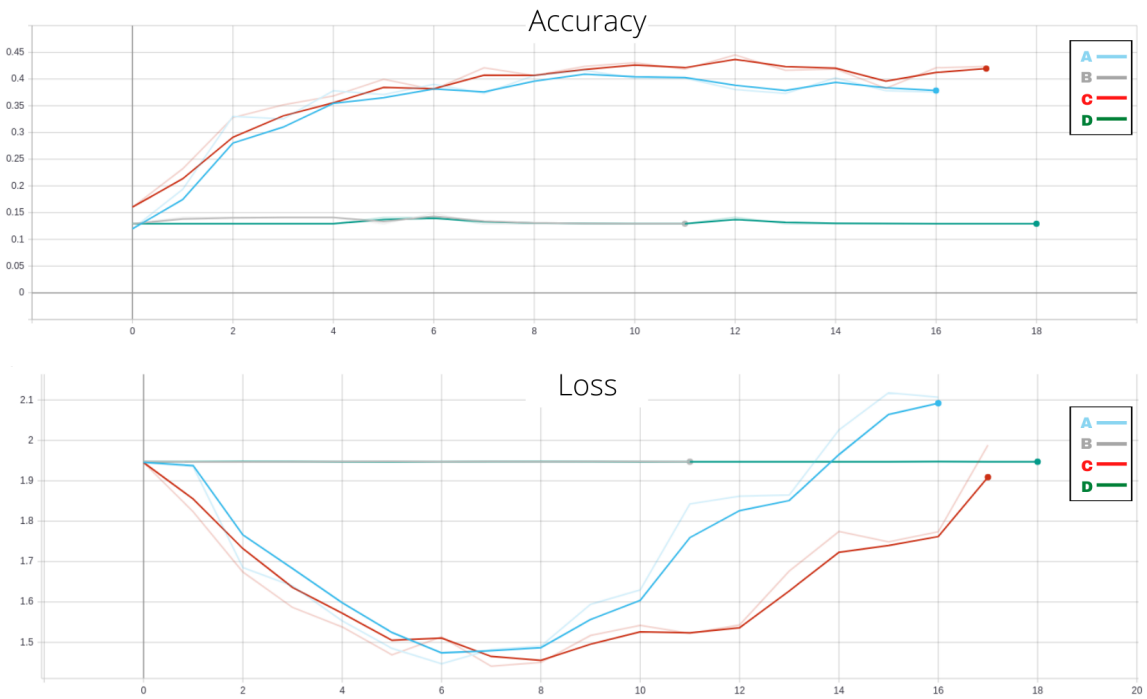

Figure 4.3: *Charts representing accuracy & loss of free Raw Dataset for each CNN configuration*

In this case as well the graph 4.3 of the accuracy and loss values for each possible configuration is shown. The values obtained are always relative to the validation of the model obtained by dividing the dataset into training part(85%) and validation part(15%).

In this situation the configurations that can be discarded are B and D since, much like in the previous case, no improvement is present during the entire training, both in terms of accuracy and loss. The cases A and C, on the other hand, are more interesting as they have accuracy values that touch 40% in one case and almost 45% in the other. The two curves are comparable, but although the difference in accuracy is almost negligible, it can be seen that the red loss curve presents a slower ascent phase.

In this case the confusion matrices for each configuration are reported as well, so we can again evaluate the precision in recognizing each individual class.

Always referring to the previous graph, we can see that even in figure 4.4 it is possible to find that CMs B and D can be ignored as they again do not provide any useful data. Once

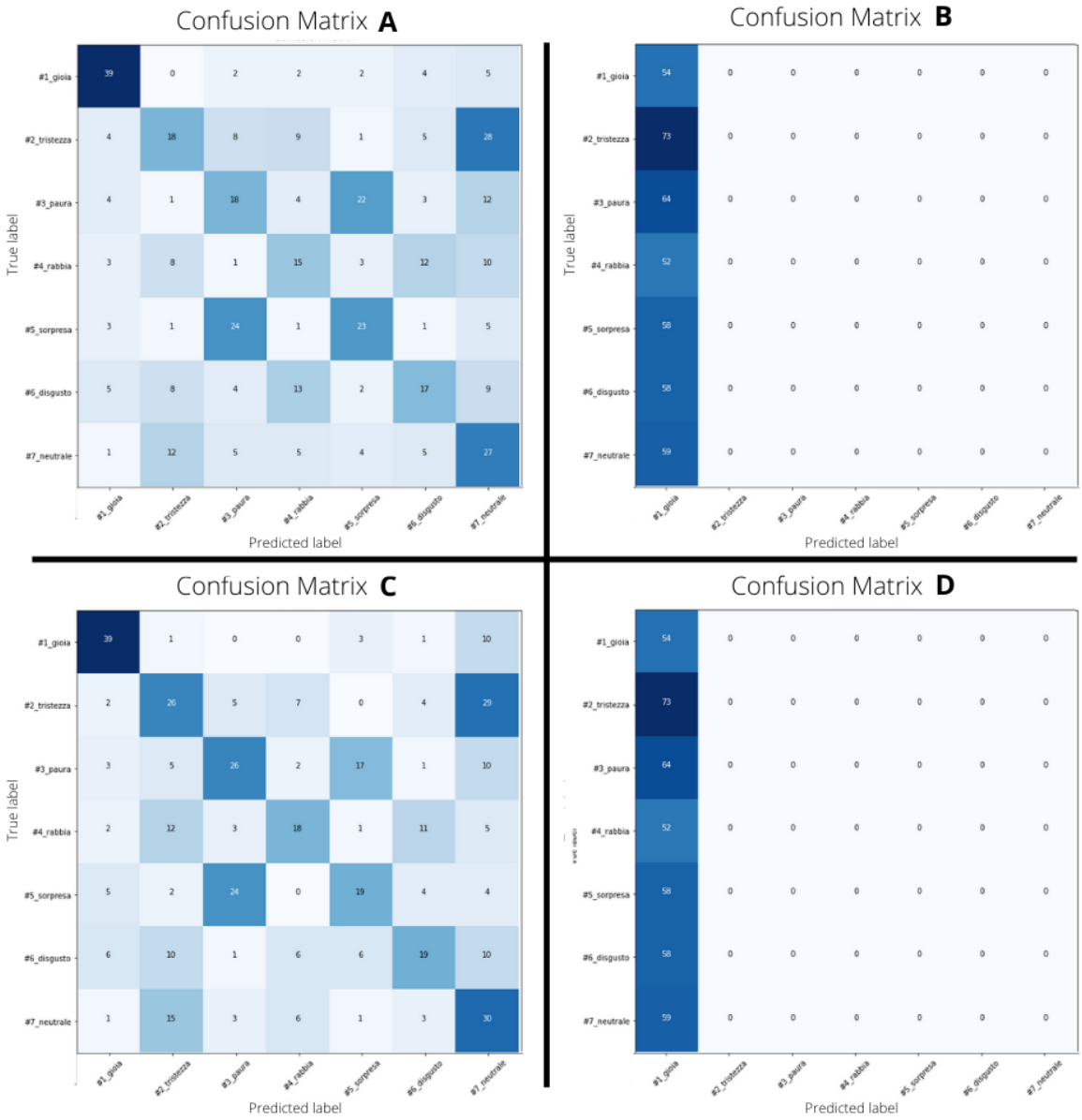

Figure 4.4: *Confusion matrices of free Raw Dataset for each CNN configuration*

again the values associated with each analyzed image are purely random.

The configurations shown on the left side contain some more information. In both cases, joy is the most recognized class, while surprise and fear often continue to be exchanged between them.

Unlike the previous case, where the landmarks were highlighted, sadness here is slightly more precise, although still often mistaken for neutrality, there is a strong decrease in the recognition of this class as fear and disgust.

Both CMs show that the dispersion tends to slightly align along the straight line that goes from top left to bottom right. This indicates an increase in the accuracy of predicting emotions. It can also be noted this assembly along the bisector is slightly more marked in the table below, indicating the subtle difference in the levels of accuracy found in the training phase and highlighted by the previous graph.

4.1.3 Modified Landmark Dataset

Now a new dataset typology is considered. By modified dataset we mean the collection of photos that have been inspected and modified, removing the images considered particularly out of context. In this situation, like in paragraph 4.1.1, the term landmark indicates that this dataset has been subjected to the normalization process where markers have been inserted to highlight important points.

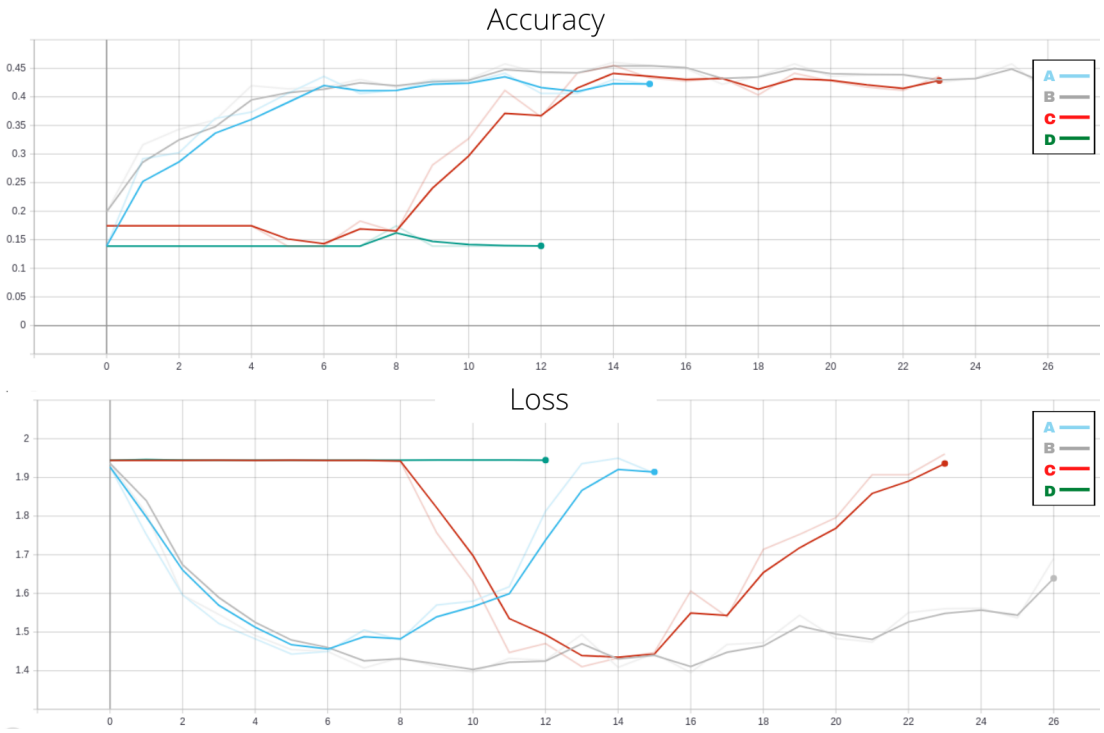

Figure 4.5: *Charts representing accuracy & loss of landmark Modified Dataset for each CNN configuration*

Once more accuracy and loss graphs regarding each single configuration are reported in figure 4.5.

The neural network structure consisting of 4 convolutions is again inadequate, in fact, even this time the green line is characterized by a linear movement mostly in the horizontal direction, without showing any evidence of learning. For what concerns the red curve, however, it is interesting to note that, compared to the others, almost a dozen epochs were necessary before seeing any improvement.

All three curves showing an upward curve reach roughly the same level of accuracy around 45%. But what you can see is the difference in the lower graph. In fact, the loss presents a considerable difference in performance between curve B and the other two, showing a really slow ascent phase and in any case only after a much greater number of cycles.

Confusion matrices for each configuration of CNN are reported in figure 4.6 even for modified landmark dataset.

We again have to exclude CM D for the same reasons as in the previous cases.

Joy remains in all the remaining 3 networks the emotions with the highest success rate.

Fear and surprise always remain very misunderstood, although with a slight improvement of the latter in case C where 33 times it is correctly predicted and only 14 times it is identified as fear.

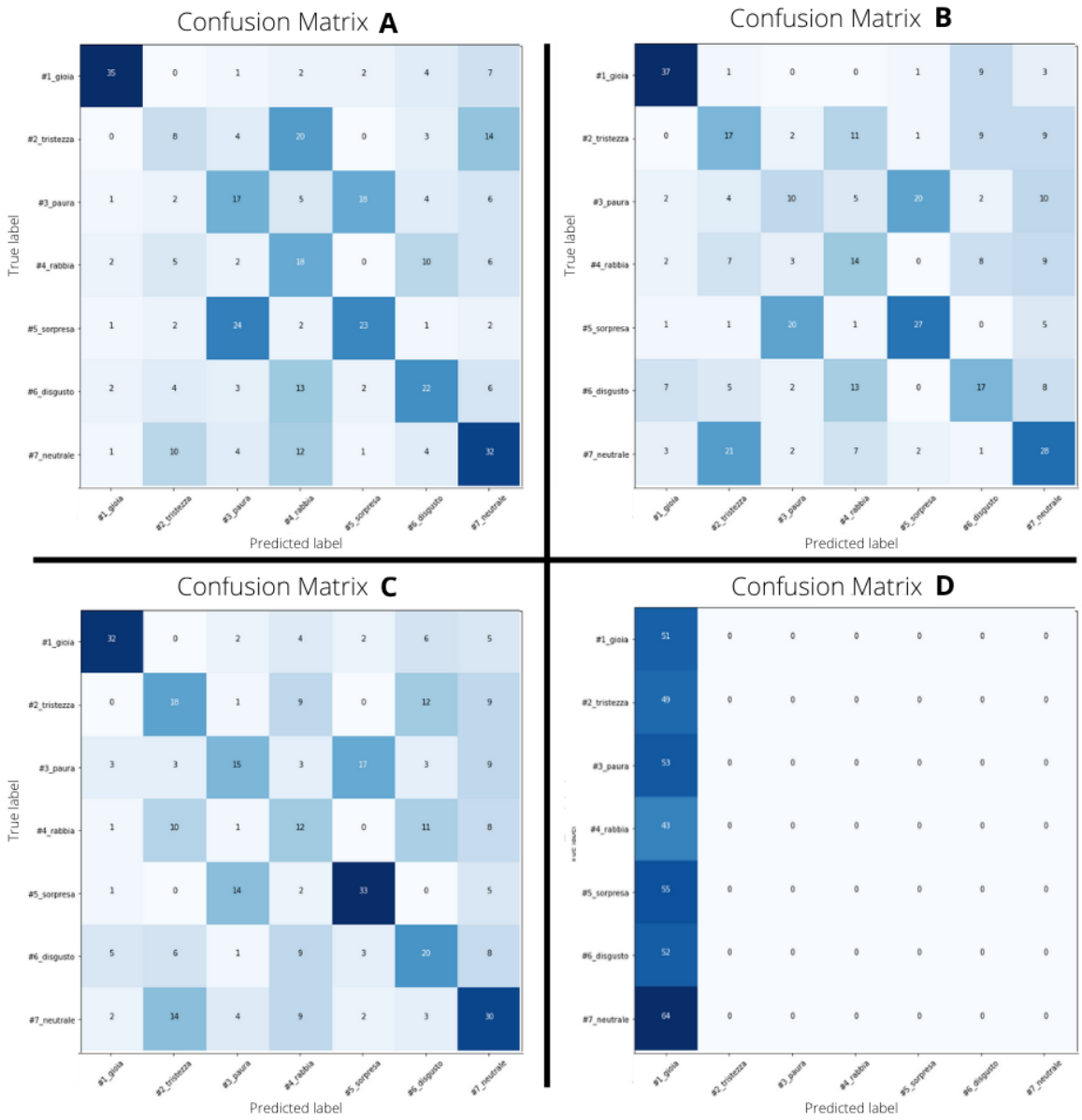

Figure 4.6: *Confusion matrices of free Modified Dataset for each CNN configuration*

In this case as well, as previously noted in section 4.1.1, sadness is the emotion with the highest error rate. Starting from CM A, where the correct predictions number is only 8 units out of the 49 analyzed, a slight improvement can be noted up to 18 correct in case C. This class presents, in any case, a particularly large dispersion, which is absolutely far away from the optimal solution. We find again that this emotion is still confused with neutral, disgust and in particular with anger.

4.1.4 Modified Free Dataset

Once again the modified dataset is used for training neural networks. This time, however, as in the case of 4.1.2, the term free means that during the normalization process no markers have been added.

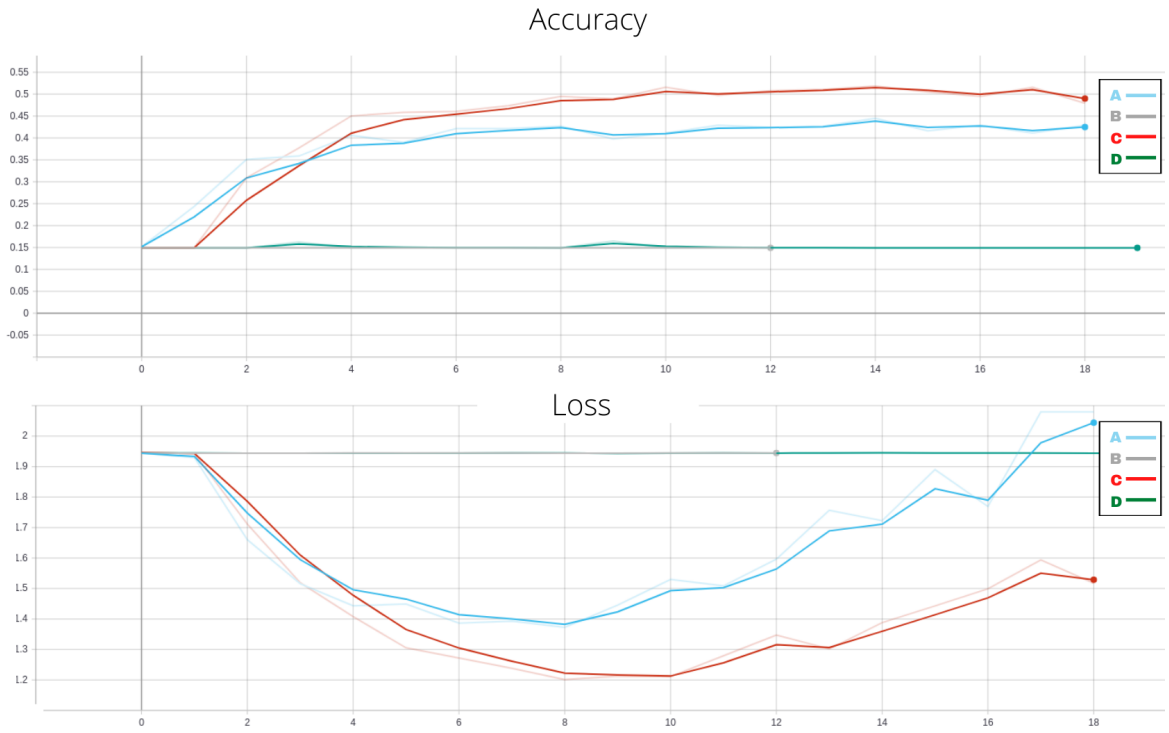

Figure 4.7: *Charts representing accuracy & loss of free Modified Dataset for each CNN configuration*

As well as in all the previous cases the configuration D did not show any sign of improvement for the entire duration of the training. The same goes for curve B shown in gray.

The red and blue curves show, as you usually expect during CNN training, curves that tend to go up in the case of accuracy, and curves that tend to go down for the loss.

In this case the configuration C shows accuracy levels between 50% and 55%, while when it comes to loss it reaches around 1.2 between the octave and the tenth epochs.

Configuration A, as can be seen from the graphs obtained with the other datasets, shows accuracy values between 40% and 45%. However, what can also be noted is that compared to other configurations, the loss curve is unable to reach the same minimum levels and that the rising edge is basically steeper, with a consequent worsening in network performance.

Again, the confusion matrices in figure 4.8 depict how accurately each class is identified at the end of the training phase, for each of the four configurations taken into consideration.

Once again, we discard the cases associated with horizontal loss and accuracy curves to analyze the remaining two.

Starting from CM A and putting aside joy, which is always the most successful data, we can note that even in this case we can notice a slight presence of a bisection line that divides the matrix into two parts. This is a good sign since the best possible result would be represented by a dispersion concentrated only along the line joining the two corners and with a minimum number of samples scattered throughout the rest of the matrix. One more aspect to be noted is a small improvement in the prediction of surprise with 27 satisfactory cases against 12 wrong ones, which however leads to a worsening in the prediction of fear, where 31 are the mistakes made against 18 positive cases only.

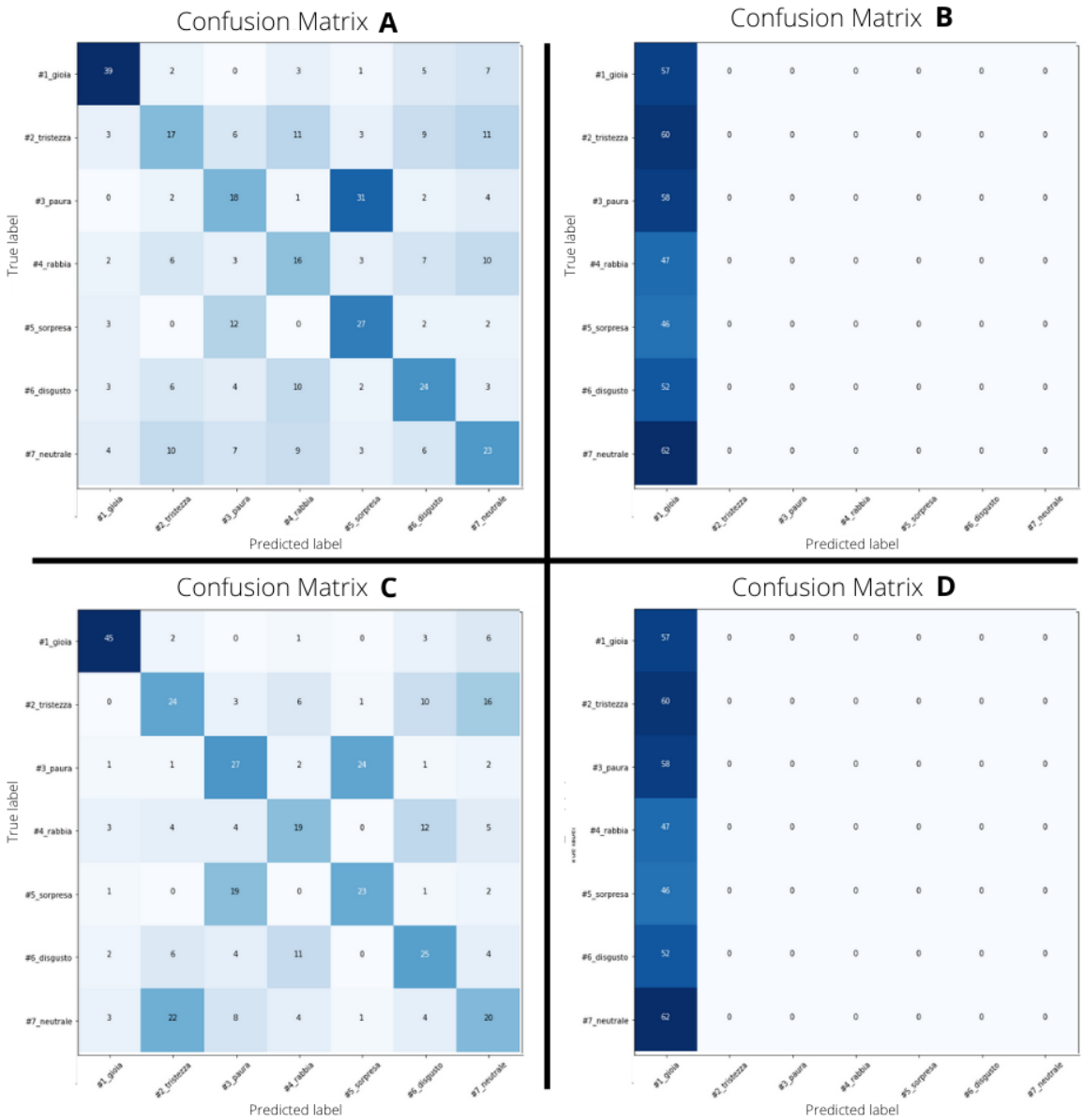

Figure 4.8: *Confusion matrices of free Modified Dataset for each CNN configuration*

Considering now the CM C, it can be seen that happiness is again easily recognizable. As regards surprise and fear, however, a slight decrease in performance at the level of surprise can be noted, combined with an improvement in terms of fear. Looking at the line of surprise on the axis of the "True label", it can be seen how 23 times it is correctly identified and 19 times it is mistaken for fear. Referring to the same reasoning for fear, it is seen that 27 are the correct predictions, 24 those in which it is switched with surprise. We can therefore conclude that these two emotions are always easily misunderstood, although with a slight propensity towards the real category of belonging.

The last consideration worth making is about the neutral class, as it is possible to see that even the prediction of this is often confused with sadness. Looking at the last line of the matrix it can be seen that 22 are the times in which neutrality has been mistaken for sadness while only 20 are the cases in which it has been correctly identified.

4.1.5 CNN Comparison

In this paragraph we compare the best result obtained from each dataset. We have again trained the best configuration for each dataset so that we can compare the graphs and try to understand what the best solution was.

Table 4.1: *CNN Comparison*

Dataset	Best Configuration
RLD	A
RFD	C
MLD	B
MFD	C

Table 4.1 shows the configuration which provide us with the best results.

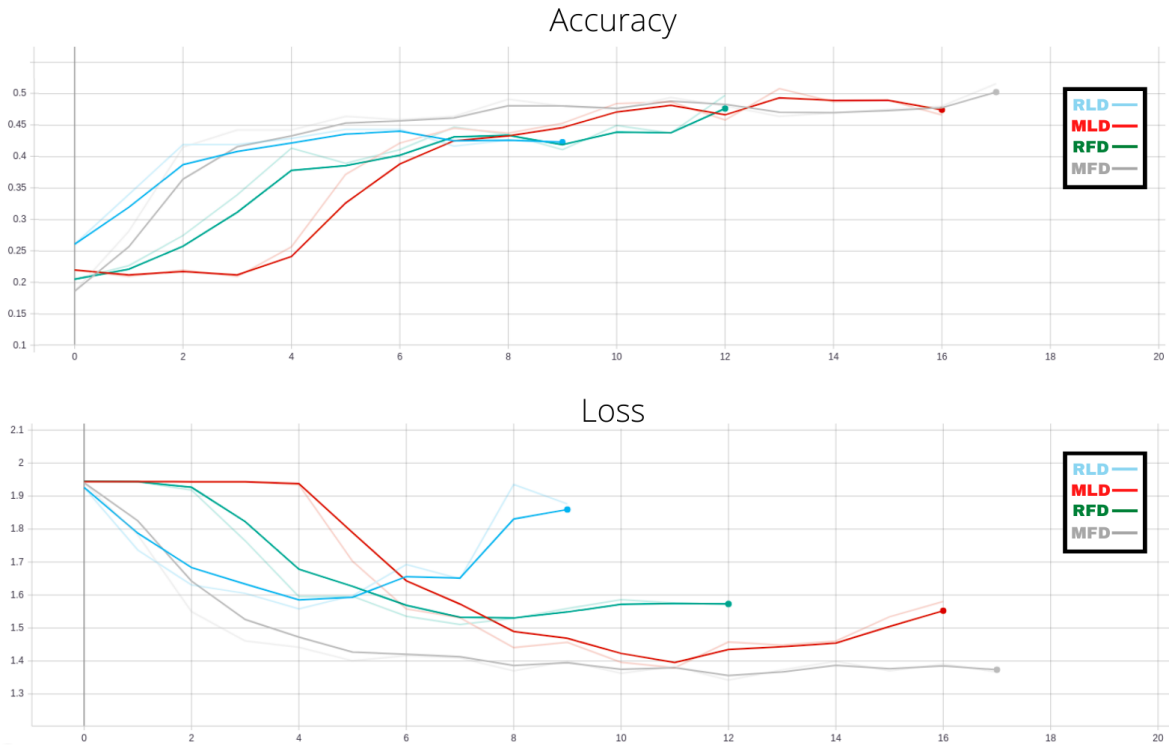

Figure 4.9: *Charts representing accuracy & loss of the best CNN configuration for each dataset*

Figure 4.9 shows the loss and accuracy graphs that allow us an easier comparison to decide which dataset and which configuration were optimal for a better project success.

As can be expected, the curves have respected what has already been described above. Although not exactly identical, the loss and accuracy values met what was expected. Note a slight increase in MLD performance and a slight deterioration in MFD.

Once again there is a too rapid ascent on the blue curve which indicates how much the configuration with only two convolutional layers is not sufficiently stable.

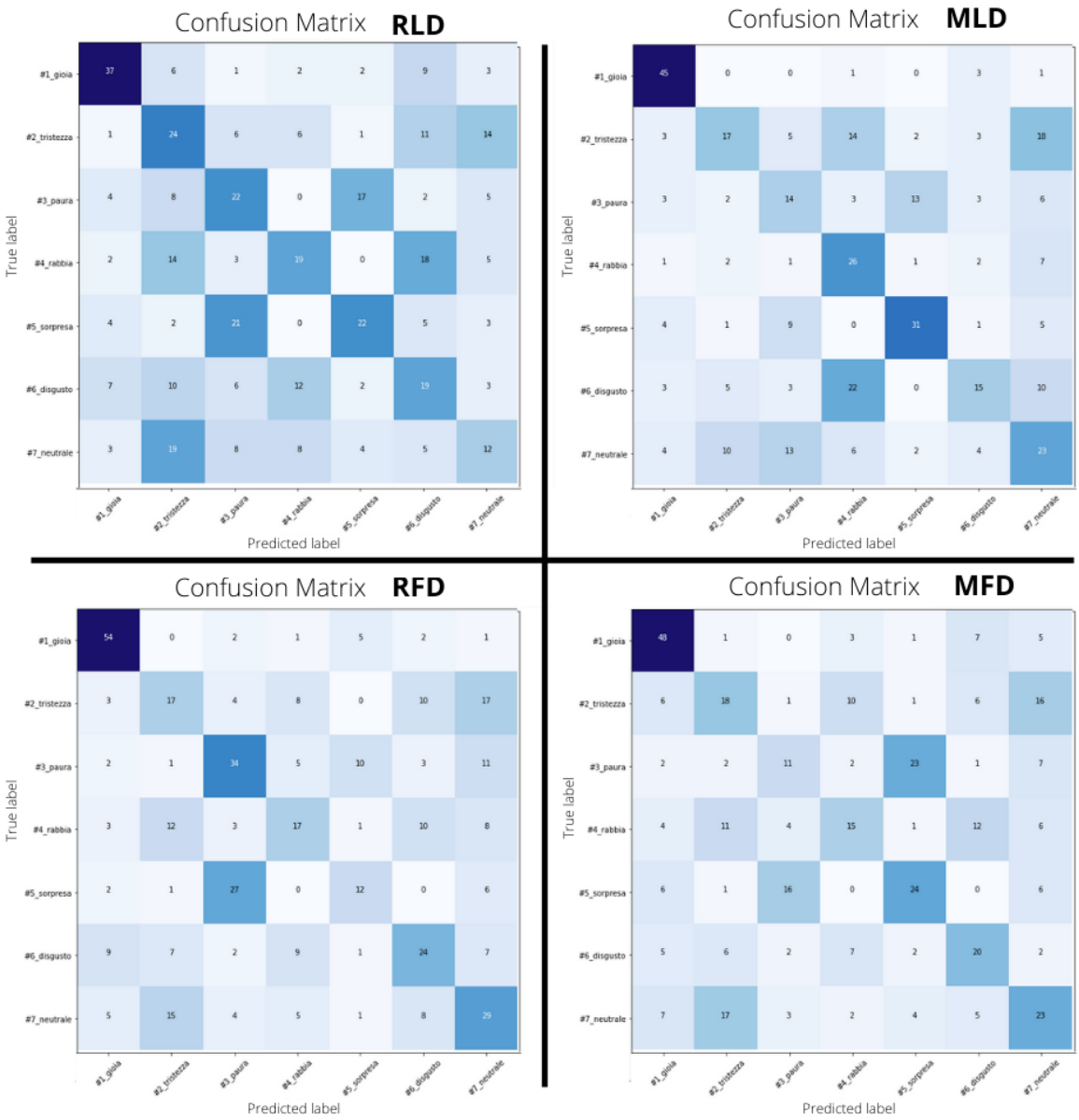

Figure 4.10: *Confusion matrices of the best CNN configuration for each dataset*

Here we can see, in addition to the general comparison between accuracy and loss, a comparison at the level of each individual class as well. Figure XXX in fact shows the CMs of which a brief comment has already been given in the previous paragraphs.

If you want to go into little more detail, however, you can address class by class. As always, joy is the most easily recognizable emotion. Sadness in almost all cases turns out to be one of the most difficult classes to predict, even if in the RLD case the results seem to be slightly more precise. On the other hand, fear and surprise are often mixed up together, as already seen previously. Anger almost like sadness is difficult to predict, although with a slight improvement in MLD where the correct cases are 26 out of 40. Disgust shows instead a good precision in the boxes below, while slight worsening is highlighted in the use of images with landmark. As for the neutral class, what we can say is that the level of precision is not very accurate because of a strong misunderstanding with sadness.

4.1.6 Augmented Dataset

This section on the performance of neural networks is dedicated to verifying the effectiveness of the augmented dataset.

Making a brief reference to what is specified in paragraph 3.1.1, with augmented dataset we mean the collection of photos formed from the dataset and the normalization algorithm that provided us with the best results (modified dataset + free normalization algorithm).

Considering that the training times for a dataset of this size are particularly long and can be prohibitive, a choice similar to the previous one has been made. It was not possible to perform training on all 4 configurations as in other cases and, consequently, we have dedicated ourselves to studying and analyzing only one configuration. Evaluating what has been described so far in this chapter, we decided to opt for configuration C, as it seemed the one that gave us slightly better results.

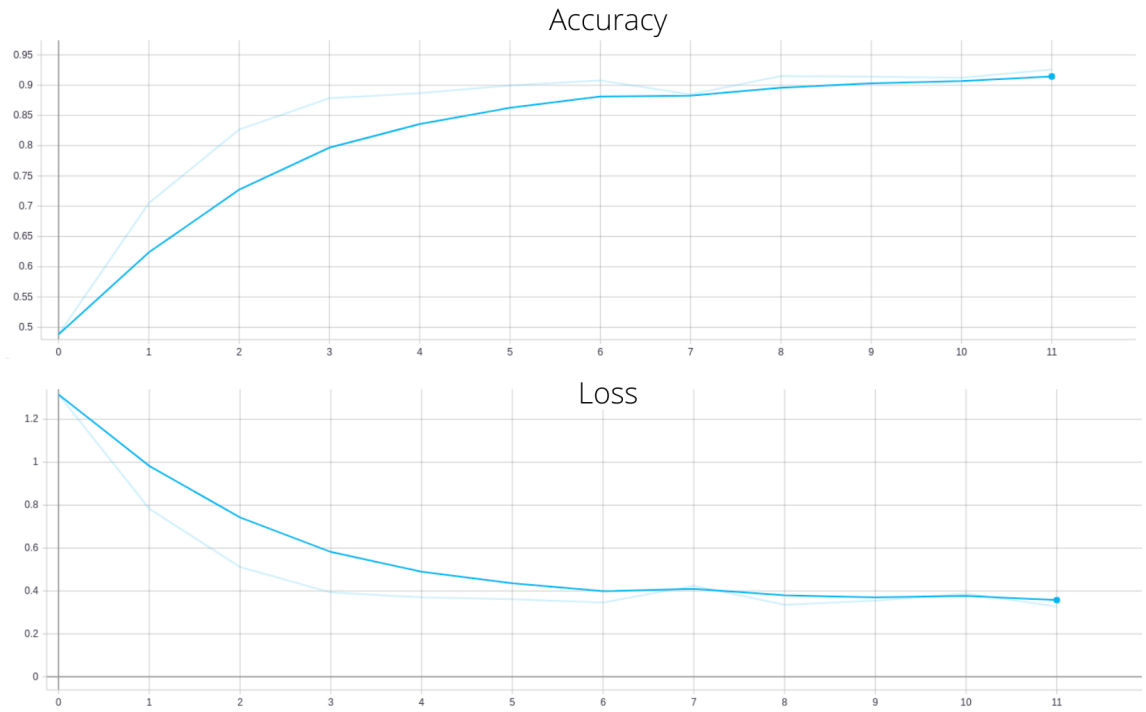

Figure 4.11: *Charts representing accuracy & loss of Augmented Dataset with validation split*

In this case as well, the curves represent the validation values obtained through a division of the dataset. 85% of the total samples used for training, 15% for validating the network.

Looking at the graph in figure 4.11, you can see how the loss and accuracy values are practically perfect. The first has a curve that reaches values below 0.4, while the second has accuracy values that even exceed the 90%.

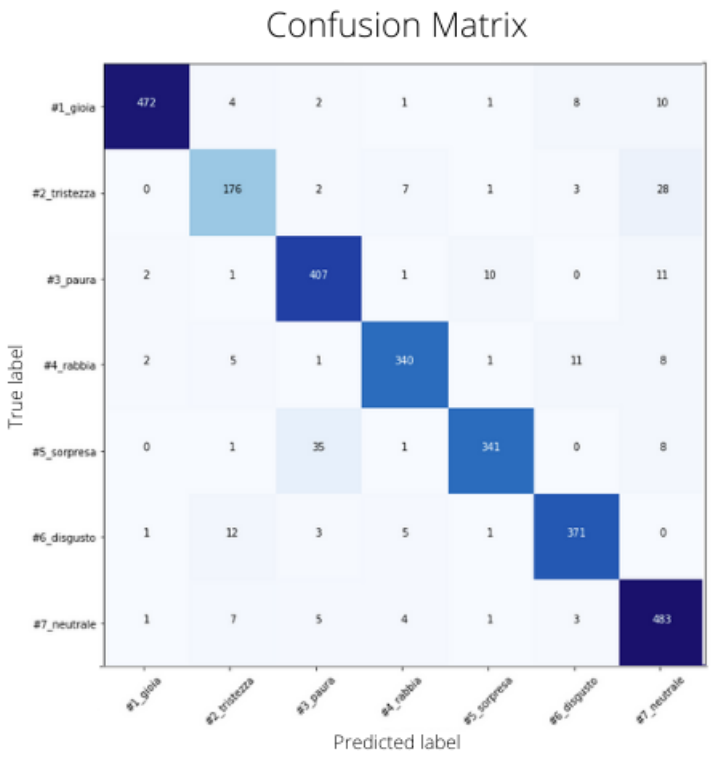

Figure 4.12: *Confusion matrix of Augmented Dataset with validation split*

If we want to evaluate these results, also divided by classes, we obtain a confusion matrix such as the one shown in figure 4.12 where the dispersion graph is almost totally linear, going to highlight almost all correct predictions.

The results shown above, although excellent, did not seem completely true and consequently we wanted to verify the effective performance of the network obtained in this way. Hence, we created a test dataset, with a few hundred samples, extrapolating some images from a dataset used in a previous thesis work [18]. In that case the analyzed classes were more than what we needed and those were images with multiple labels. Given these differences, not all the dataset could be analysed, but we preferred to manually extract some useful photos to verify the results of our network.

The graph in figure 4.13 shows the validation curves carried out on the 367 images enclosed in the test dataset, following the training through augmented dataset.

What can be easily seen is that the accuracy values do not absolutely reflect those found in figure 4.11, but appear to be close to the values obtained in the first four analyzed datasets. This value is in fact around 40/45%. Although the accuracy greatly differs from the previous graph, the one that most attracts attention is the loss curve. In fact, unlike all the other cases analyzed, this does not show any descent interval, but has a steadily increasing development, until it reaches a value never reached so far.

Once again, what is reported in the accuracy and loss graphs can also be noticed in the confusion matrix level in figure 4.14. Here you can see how the desperation is far from being linear and indeed this is highly heterogeneous.

Finally, by comparing the data obtained with the augmented dataset we can notice a strong disparity in the results.

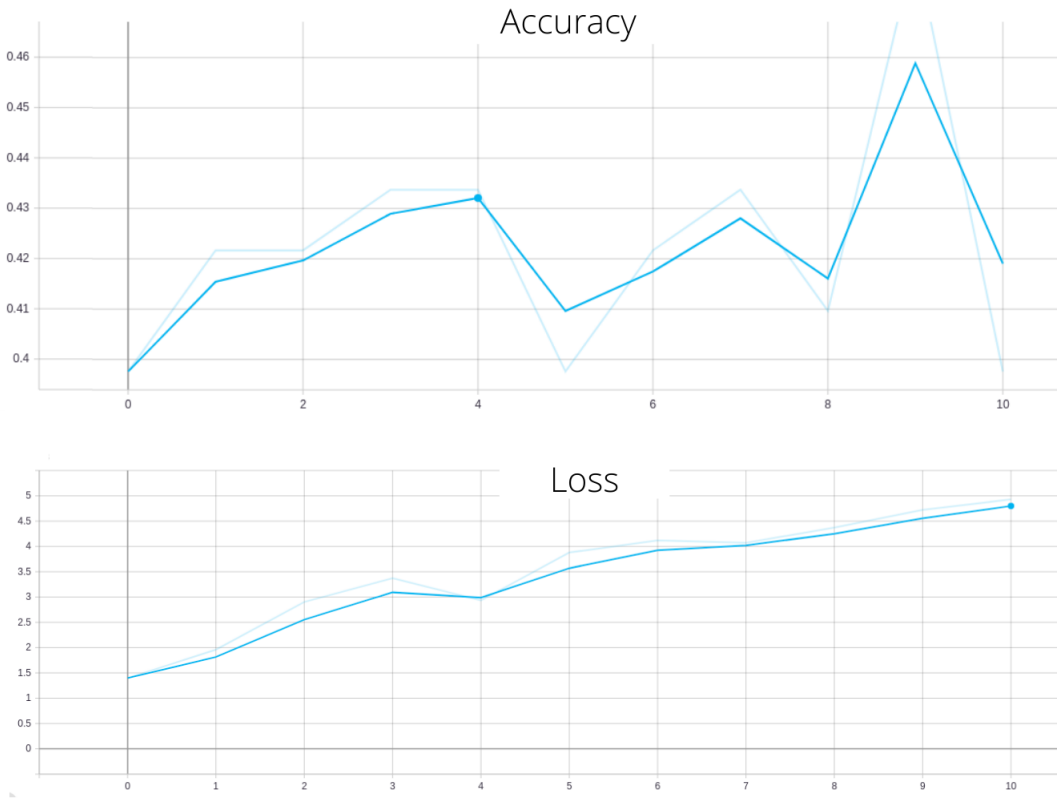

Figure 4.13: *Charts representing accuracy & loss of Augmented Dataset with Test Dataset validation*

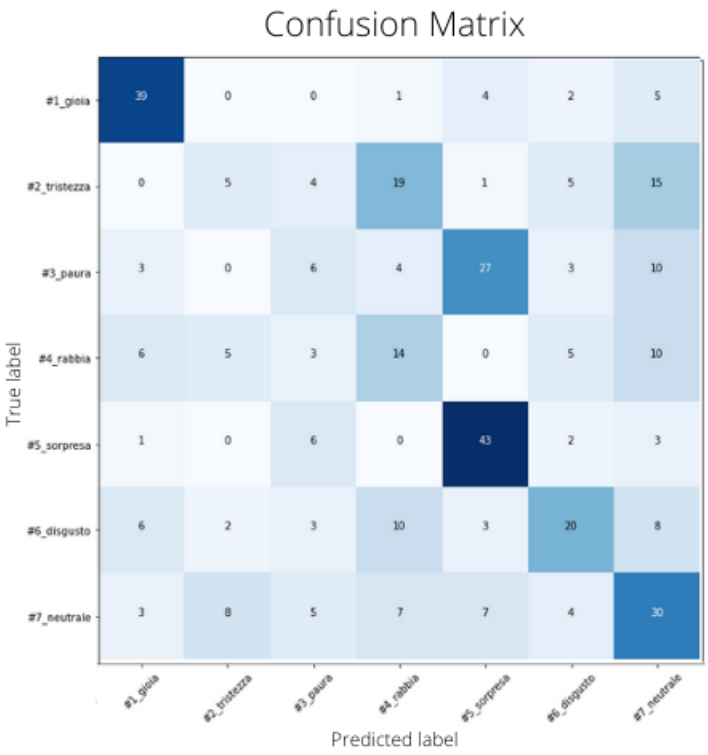

Figure 4.14: *Confusion matrix of Augmented Dataset with Test Dataset validation*

4.2 Speed Tests

Before going into detail on the results obtained on our device and our network, we believe it is appropriate to show some time values regarding the advantages offered by the use of the TPU present on the new technologies offered by Google. For completeness, in addition to the GCDB, the data referring to the Accelerator, another Google device equipped exclusively with TPU and which needs to be physically connected to the PC via USB, are reported as well.

The Table 4.2 compares the time spent to perform a single inference with some example models on the Edge TPU. For the sake of comparison, all models running on both CPU and Edge TPU are the TensorFlow Lite versions.[1]. These data measure the time required to execute the model only. These tests are also performed using C++ benchmark tests, whereas Python benchmark scripts may be slower due to overhead from Python.

Table 4.2: *Time per inference, in milliseconds (ms)*

Model architecture	Embedded CPU (Dev Board)	Edge TPU (Dev Board)	Desktop CPU	Desktop CPU & Accelerator (USB 3.0) (EdgeTPU)
DenseNet (224x224)	380	20	1032	25
MobileNet v1(224x224)	53	2.4	164	2.4
MobileNet v2(224x224)	51	2.6	122	2.6

The test is interesting when you compare the computational time of each different hardware component. These results, taken from the official Coral Dev website, are based on the type and the speed of each technology and can vary according to a variety of factors. On the website it is clearly specified how an individual TPU is capable of performing 4 trillion operations (tera-operations) per second (TOPS), using only 0.5 Watt, presenting all the power this hardware can guarantee.

What deserves to be highlighted is that the table is not shown to compare the very little differences between the different models, but to present the really big differences between columns which indicate how much faster edge TPU inferences are compared with the CPU versions

This consideration is very important in order to underline the advantages in the use of an Edge TPU compared to a CPU and to understand the reasons why we choose one technology over the other for better success in reaching our goal.

The same performance analysis, in terms of time, was carried out to evaluate the model, comparing the results obtained using the GCDB and those obtained using the CPU of our laptop.

[1] `https://coral.ai/docs/edgetpu/benchmarks/`

4.2.1 Software version

What we call 'Software Version' is the code version, developed to run on a CPU installed on a Laptop. The PC used for the test is equipped with a commercial CPU Intel Core i7-8565U whose specifics are reported in Table 4.3.

In this case, both TensorFlow model and main PY code run taking complete advantage of the computational power of the CPU. Therefore it is not necessary to convert the model in TF lite version, but to run the original one without the need of the Interpreter and exploiting all the Keras functions included in Tensorflow libraries.

Table 4.3: *Intel Core i7-8565U*

Core	4
Threads	8
Processor Frequency	1,80 GHz
Max Turbo Frequency	4,60 GHz
Cache	8 MB Intel Smart Cache

4.2.2 Coral Dev with Edge TPU

The Google Coral Dev is structured as in table 4.4 ARM architectures are a family of 32-bit microprocessors generally used as micro-controllers in embedded systems. Due to their low electric power consumes related to the performances, these kind of architectures overlook the mobile devices' market where energetic saving is fundamental.

Table 4.4: *Google Coral Dev*

Main system-on-chip	
NXP i.MX 8M SoC	Quad symmetric ARM Cortex-A53 processors at 1.5GHz Support of 64-bit Armv8-A architecture
ML accelerator	
Edge TPU coprocessor	ASIC designed by Google that provides high performance ML inferencing for TensorFlow Lite models 4 TOPS (trillion operations per second) 2 TOPS per watt
Memory and storage	
Random access memory (SDRAM)	1 GB LPDDR4 SDRAM 1600 MHz maximum DDR clock

The GCDB, as it has been already seen, takes advantages of the Edge TPU technology and allows fast computational times to run Neural networks, previously converted in TF

Lite. So we can call this the "firmware" version, because all the main code is adapted to work together with the converted model. Coral Dev is also developed with a CPU installed on it, which is in any case the main processor of the device and is the one where to install the "firmware".

4.2.3 CPU vs TPU

The comparison between CPU usage and TPU usage, in terms of speed, is done computing the time of the main processes in table 4.5:

- Initialization Time: the time the program takes to initialize the model, the algorithms and the video stream from the camera.
- Time for one Cycle: the time required until each cycle is over, starting from the first face recognition process, in order to calculate all the features and emotions. It ends when all emotions are extracted and the collected and classified data are sent.

Table 4.5: *Computational Time*

	CPU	**TPU**
Initialization (*s*)	7-8	10-11
For One Cycle (*s*)	~ 0.3	~ 1.3

The delay in streaming the video is another aspect of particular importance for the success of the project and it provides a good comparison between devices. The stream delay is computed from the moment the camera captures a frame until the moment that frame is visible in the program; hence, it is the latency in sending frames from the camera to the device, which changes depending on the processes executed:

- when people are not in the camera view and recognition processes do not start, the latency is lower
- when people are in the camera view, recognition processes start, the latency is higher

Table 4.6: *Video Stream Latency*

	CPU	**TPU**
Without Processes (*s*)	0.5 - 1	1.5 - 2
With Processes (*s*)	2 - 2.5	3.5 - 4

Note:*The Delay of the stream is mainly due to the WiFi connection and its speed conditions, which affect the speed in which the frames of the video are transmitted. Tests are conducted in the same WiFi conditions, the same distance from the router is maintained and in presence of high connection band, so that the latency due to the connection could be considered equivalent for each devices. Changing WiFi connection and modifying the previously described conditions, the values reported in table 4.6 could vary.*

Chapter 5

Discussion & Conclusion

5.1 Discussion

5.1.1 Convolutional Neural Networks

During this thesis project several attempts to train different neural networks were made. Different configurations to be analyzed have been created and various datasets have been used for the training phase.

As reported in chapter 4 for each of the datasets created, four different types of CNN have been tested. Each of these has provided different results and some considerations can be made.

The first thing that immediately catches the eye is that the configuration made up of 4 convolutional layers proved to be absolutely unsatisfactory regardless of the type of dataset used for the training phase. It can be seen that in all cases, loss and accuracy graphs show no improvement for the entire duration of this period. We therefore deduced that the use of four convolutional layers is not suitable for analyzing the type of images we had and that consequently this neural network structure could be discarded.

Another thing to underline is that configuration B, almost equal to what was said for D, can be neglected. In fact, it proved unsuccessful in three out of four cases. The only case in which this structure proved to be satisfactory was that in which the training took place via modified landmark dataset. Using this dataset, as can be seen in figure 4.5, the results were found to be better than others, so much so that for the section where the various datasets were compared, this type was chosen.

As for the network that has given the least problems considering all the analyzed datasets, we can say that it is the one made up of only two convolutions states. When checking all the graphs, the only configuration that has shown signs of learning, regardless of the input used, is identified by the blue curve and consequently by structure A. This, however, in four out of four cases has loss curves with too rapid trends and with too steep rising edges, which indicates an unstable and non-reliable network.

As for the 3-layer convolutional structure indicated with the letter C, it can be seen in table 4.1 that in two out of four cases it was selected as the most suitable configuration. Paying a little more attention you can also see a detail that can be interesting: both cases in which this structure has proved to be the most suitable are identified by normalization without landmark.

If we want to go deeper into the topic, some analysis can be done on what the best dataset and the most suitable normalization algorithm could be. Let's go back to look at the graph in figure 4.9

What can be seen from this graph is that the red and gray curves show better accuracy and loss values than the blue and green ones. This indicates that the work of removing the images deemed unsuitable has led to an improvement in the performance of the model. This data therefore suggests that, when creating a dataset, it is really important that all inputs observe the characteristics of the class they belong to. In this case, it would be interesting to further modify the current datasets, removing the remaining borderline cases to verify if this would lead to a further improvement in the effective learning of the network. Unfortunately, however, for reasons of time, it will not be possible to verify this hypothesis in this thesis project.

As regards the type of normalization carried out, referring to the same figure, it can be noted that starting from the same dataset no big differences in accuracy are evident comparing the two normalization configuration. In fact, comparing the green curve with the blue one and the red one with the gray one, we can see how the accuracy levels vary only by a few units in percentage. Instead, what is slightly more evident appears in the loss graph. Although at absolute levels the difference is minimal, the shape of the curves indicates instead that the images processed with the addition of the landmarks have steeper rising fronts. As regards the curve linked to the RLD, this data can be amplified by the use of a network with only two convolutional layers which, as already mentioned, are subject to this type of trend. But the same thing cannot be valid for MLD, as both this and MFD are obtained by training neural networks made up of three convolutional layers.

With these considerations, we can therefore deduce that the modified dataset is slightly better than the raw dataset and that the addition of markers to highlight key points on people's faces does not present particular benefits, on the contrary it leads to a slight deterioration in the loss graph.

Finally we can draw that the combination that has guaranteed us a better result is as follows:

1. Modified Dataset
2. Free Normalization Algorithm
3. Neural Network with three convolutional layers, C configuration

Once these results were obtained, we wanted to try one more step to verify if it was to reach values that would guarantee better model performance. For this reason we used the modified dataset to obtain the increased one. As specified in chapter 3, to form this dataset, images' brightness, depth and dimension in the 2D space have been modified. This meant that the same image was repeated several times, modified only in one or more parameters simultaneously. Once the increase of the number of samples was completed, the normalization algorithm without the addition of markers was carried out and the images were processed by the neural network to obtain the values in figure 4.11. As in all the previous cases, the values shown here represent the validation curves obtained with a split of the dataset. In figure 4.13 the same network and the same dataset have been analyzed, but the validation curves have been obtained through the use of a dataset created specifically to verify the real effectiveness of the model.

Comparing the two graphs it becomes evident that the results are particularly different, indicating untruthful values in at least one of the two cases. By checking the augmented dataset a little more carefully, we note that the images used for the training of this network are not as manifold as expected. Many of these, in fact, are repeated the same way without any changes, especially those modified in brightness. We therefore note that following the

application of the normalization algorithm, the images that had initially been modified appear to return almost completely to their original state.

By applying a division to a dataset composed in this way it is easy to understand that many images used for training the model are very similar, if not totally identical, to those used to validate it. Hence the optimal performances presented in the graph.

On the other hand, the data obtained through validation with the test dataset show an exclusively increasing loss curve, highlighting that the repetition of the same image several times in the same training cycle limits the ability to generalize the model, severely limiting its ability to recognize emotions on new faces.

Finally, we realized that the use of algorithms to increase the size of the dataset was not particularly useful in our case. In fact, despite the good results during the training phase, we have obtained not so acceptable results testing the network with another dataset.

It is maybe possible to improve outcomes executing the augmentation only after having already done the normalization processes and changing some parameters of the augmentation algorithms to eliminate the most of identical images and intensify the modified characteristics. We therefore opted to discard this choice and to focus on the data obtained through the combination described above.

5.1.2 Devices

One of the aims of the project is to take advantage of a device able to improve and accelerate the emotion analysis by an Artificial Neural Network. In literature it is not easy to find a good way to reach this goal and there are not so many devices made for Deep Learning.

As we have previously seen, in Chapter 4.2, GCDB is one of the best solution to run a Neural Network: the on-board Edge TPU is capable of performing rather complex networks in a power-efficient way and it is clear that Neural Networks are always faster when run on a TPU, as shown in Table 4.2. The use of the Edge TPU in the Coral Dev substantially improves the performances and reduces the execution time for all the networks.

These differences in terms of time and performance pushed us to compare the results both in the Google Coral Dev version and in the software version running on the PC.

The software version was the first one to be created, since it is essentially the original version of the code and the original version of the TF model from which the others are extracted from. Indeed, the TF lite model and the embedded devices compatible version are just a translation or a modification of the original one.

Having this double possibility, we decided to compare the two to demonstrate the real advantages in executing the entire work with a device as Coral Dev, rather then just using a computer.

As we can see from the Table 4.5 in Chapter 4, the execution times for the CPU and the embedded Coral Dev system analysis are quite different. The results show that, in the Coral Dev, the processes are slower than the one on PCs, even thought the CGDB model runs completely on the Edge TPU. At first this information seemed to be unusual, since theoretically Edge TPU model performances should be higher. With a more careful analysis, on the other hand, we understood that the time indicated in our tables is not linked to the time of execution of the model only, but to the entire analysis process as well, starting from the localization of the faces up to the emotional recognition.

We noticed that the execution of the entire procedure is made up in very heavy part of image processing which has greatly influenced the results obtained. Therefore all the advantages obtained from exploiting the Edge TPU for running the model are lost due to the slowness in carrying out the preprocessing phase by the CPU of the embedded device.

The model we created is not extremely complex and the fact that it processes only one single image per time ensures that the advantages obtained for its processing at the TPU level are compensated by the delay in the preprocessing phase caused by a less performing CPU on board.

Considering all the steps of the face recognition process, such as face recognition and age and gender detection, we can say that this part has a stronger influence than the real emotion detection on the final performance of the system.

We can therefore conclude that the results of the whole process are better in the Software version, although the model is not optimized yet in the TF lite version.

Another fundamental characteristic to keep in consideration is the latency time of the video stream. Unlikely this parameter changes depending on the different cases the software is running. Looking at table 4.6 we can find that the lowest values of delay are shown when no faces are framed into the camera and the images don't need do be elaborated. On the other hand the highest latency is when the program runs all the face and emotion recognition processes. This is probably because of the higher computational power required to perform all the passages which cause an accumulation of a little more delay in showing us the frame.

This fact is important since the moment the frame is shown is also the moment when all the analysis are ended and data are collected and sent, so if latency grows up, the time Pepper is still waiting from data requests increases as well.
Despite this considerations, it is noticeable from Table 4.6, that the stream delay during the execution of recognition processes is, in both devices, about two times the delay without that processes execution.

Focusing on the results, it's clear that, generally speaking, the performances of the computer version are slightly better than the Coral Dev version and this is certainly due to the differences between the technology of the laptop compared with the one of the board:

- The Edge TPU technology is a powerful tool for deep learning analysis, but obviously the CPU integrated on the board is not sufficient to elaborate the huge amount of data, required for our preprocessing part, in the same times of i7 commercial technology, which is able to also run a structured Neural Network without being subjected to perceptible slowing down.

- The same considerations can be made about the video stream and the WiFi transmission: the Internal WiFi Network Card on the laptop is better in performance than the one on the board so it is comprehensible on the Coral Dev to have the stream of the video being about two times slower, despite the same WiFi configurations.

The Coral Dev shows up a pretty big amount of advantages such as the power in relation to the price and dimensions, its portability and the possibility to keep the device always on, thanks to its efficient consumes, without the need to initialize the program every time. At the same time, the small latency while extracting the data is negligible for the only aim of data storage and analytic, but it is not fully sufficient in a real time application, like the one discussed in this project. Therefore this delay can cause a misleading reception of the right emotion by Pepper causing an erratic conversation, while it is interacting with a person. If we think about a real case, when Pepper makes a request to the server to receive the data containing the emotion, the latter will not represent the emotion of the subject at the current moment, but the one that he or she had a few moments before. Through this delay therefore there is a risk of providing the robot with an incorrect emotional reading, since when it requires the emotion, the image that is analyzed contains the expression that the subject showed a few seconds earlier.

This issue could be solved introducing a delay also in Pepper BMS in order to compensate the latency and have in this way a correct predictions. Although this solution is functional, it provokes a loss in fluency of Pepper conversation but we can consider it negligible because it is about only few seconds.

Some possible alternative solution could be investigated.

Examining all the results, one of the best technical solution would certainly be the integration of the CPU system with an external TPU such as the solution offered by the USB Accelerator we have briefly mentioned above, which is a TPU-integrated device, directly connected to the laptop. Its aim is to split the process in two parts, running all the preprocessing steps on the CPU while the only emotion recognition analysis is performed on the Accelerator. Obviously, despite of the probable improvements in terms of speed and computation, all the practical and comfort advantages of using a board like Coral Dev would be lost.

Another possible solution could be to integrate all the preprocessing and people recognition steps directly into the robot's system and send the neural network, uploaded to the device, only the data useful for the emotion analysis, therefore using the robot's hardware component and the camera integrated in it, limiting in this way the delay due to video frames transmission, entrusting real time emotion analysis exclusively to the Edge TPU power.

5.2 Expectation Verification

1. The Development of CNN able to perform adequate predictions: the results are quite good, although they can be further improved by training the model with a more efficient dataset, with the addition of large number of photos or upgrading the size with augmentation techniques. In fact, the results are worse than what we expected from literature, but also the limitations in the network structure due the mandatory conversion in an Edge TPU compatible model can be a partial cause of this partially suitable results. No other information about a system like the one described in this thesis project are provided so far and so a strict comparison cannot be performed.

2. The Development of an algorithm for real time processing of visual information is successful, the integrated code is usable for different type of application and situations. We kept it as general as possible to make it suitable both to work using the integrated camera of the robot and to be connect to every single camera with an internet connection.

3. The previous two points are, then, integrated in the development of an advanced stand alone system optimized to FER analysis, the Google Coral Dev, with positive results, although the device is of new generation and its performances could surely be improved.

4. Another reached goal is the storage of the results in a database for profiling subjects interacting with the robot, through the use of an ad-hoc developed server which allows communication between the device, the robot and MongoDB, a database where the data is saved and stored.

5. The Increasing of Pepper's communicative ability and empathetic skills is the less successful result for two main reason. The first is certainly the poor precision in prediction. The second is the delay in the video streaming which could cause errors in the interpretation of the emotional state in real time, leading to an erroneous response from the robot. To avoid this second issue, a delay in Pepper BMS is added causing a worsening in his communication skills

5.2.1 Work Limitations

During the thesis work, were found some factors that partially limited the potential of the project.

The first thing that can be highlighted is how much the creation of a customized dataset capable of providing excellent results is a very difficult task. Firstly, due to the fact that in this specific case the availability of the subjects has been limited, although we have tried in every way to expand our range to the maximum. In second instance, the samples collected do not represent expressions of real emotions. All this makes us understand how important data is: both in terms of quantity, but even more in terms of quality.

Another constraint is linked to the creation of the network structure, as, as anticipated in chapter 3, today there are still many limitations imposed by the producers of the GCDB, to ensure that the model can be correctly mapped on the TPU. We were forced to use a sequential model, although in the literature some networks with branches in parallel had been evaluated and had shown good results. By expanding the possibilities of mapping an increasingly complex network configuration at the TPU level, the performance of the entire device could be improved.

A further limitation is connected to the heavy preprocessing phase that must be carried out in order to be able to analyze the video in real-time, to process it to prepare the image making it suitable for being analyzed by the model and finally to obtain a result in terms of recognized emotion. All these steps require a fairly high computational power, so that they risks not being suitable for execution on the CPU of the embedded device. In fact, from table 4.6 it can be seen how the delay times found in the video streaming can reach values that are not completely negligible considering the final purpose of the project. To overcome this issue, one solution could be that of carrying out the preprocessing phase by exploiting a CPU more powerful than that supplied by the GCDB and of using this device for emotional analysis only.

This solution immediately made us think of directly exploiting the internal CPU of the robot, in order to manage the video directly inside Pepper and connect it to the external device only during the final analysis of the data. This idea, however, clashed with the fact that the hardware available to the robot is already widely used for its basic functions and it was not possible to weigh it down further.

5.3 Future Developments

1. Find different solutions for the dataset choice or completion:
    - evaluate the possibility to modify preexisting datasets in order to obtain a photo classification suitable for the emotion recognition and Neural Network analysis
    - find other faster solution to complete the number of data to be collected or an augmentation algorithm in order to grow the dataset
2. evaluate future TF Lite packages' upgrades that will improve the number of Tensorflow's functions and layers could be successfully converted and mapped
3. improve the interconnection of the entire full system, in particular between robot and device, evaluating new devices will be developed or using new technologies in robot's hardware or for deep learning analysis.

5.4 Conclusion

The final result showed us that when the work was completed, the robot partially learned to recognize emotions, in fact the probability of success is around 50/55%. These values in general terms do not make the robot able to manage a conversation based on this specific analysis in full autonomy. The margin of error is still too large and the risk of comments or sentences out of context, due to an inaccurate analysis, may prove to be a worse choice than the usual impersonal speeches of the robot. In any case, not all the work was unsuitable since with an adequate statistical analysis it is possible to obtain all the data of interest regarding the aspect of profiling of the interlocutors. Also when it comes to the management of Pepper's empathetic abilities, some important observations can be made: the value of general accuracy is an average value that takes into account all possible emotional states, but as we saw in the results chapter, each class has different probability success. What can be done in this case is therefore to limit the robot's communication skills to the basic ones when the data obtained belongs to an unreliable category, and on the contrary manage the most promising classes in order to leave the robot more freedom of expression with consequent possibility of creating a more personal contact with the subject thus increasing his empathetic skills.

Wanting to make a further evaluation on the project we can say that the choice to create a new dataset from scratch has proved to be a more complicated choice than expected. We would have expected to get a lot more data and a higher involvement in order to obtain a complete dataset able to provide largely satisfactory results. However, what we have found is that the collection of data is a very complex task and that to be able to execute it properly it would have required much more time and many more resources. All of this gave us the opportunity to understand how valuable it is to be in possession of valid data.

In conclusion, this project has given us the opportunity to interface with two of the most innovative technologies to date and which still have strong margins for improvement. It allowed us to evaluate all the positive and negative aspects connected to these and to test ourselves by trying to build a bridge between the two realities, exploiting the potential of one to improve the performance of the other with the aim of creating a robot capable of perceiving the feeling of the subject and of interacting with them in the most "human" way possible.

Bibliography

[1] Edward A. Feigenbaum Avron Barr. *The Handbook of Artificial Intelligence*, volume Volume 2th. Department of Computer Science, Stanford University, 1982.

[2] Ravi Kalakota Thomas Davenport. The potential for artificial intelligence in healthcare. *Furure Healthcare journal*, 2019.

[3] Liu Yang Tian Wu Yawen Li, Weifeng Jiang. On neural networks and learning systems for business computing. *Neurocomputing*, 2017.

[4] A guide to deep learning in healthcare. `https://www.nature.com/articles/s41591-018-0316-z`, 2019. Naturemedicine.

[5] Paul A Beach, Jonathan T Huck, Melodie M Miranda, Kevin T Foley, and Andrea C Bozoki. Effects of alzheimer disease on the facial expression of pain. *The Clinical journal of pain*, 2016.

[6] Laura Alonso-Recio, Juan M Serrano, and Pilar Martín. Selective attention and facial expression recognition in patients with parkinson's disease. *Archives of clinical neuropsychology*, 2014.

[7] Joseph Manfredonia, Abigail Bangerter, Nikolay V Manyakov, Seth Ness, David Lewin, Skalkin, et al. Automatic recognition of posed facial expression of emotion in individuals with autism spectrum disorder. *Journal of autism and developmental disorders*, 49, 2019.

[8] Jacenta D Abbott, Tissa Wijeratne, Andrew Hughes, Diana Perre, and Annukka K Lindell. The perception of positive and negative facial expressions by unilateral stroke patients. *Brain and cognition*, 2014.

[9] Mira Jeong; Byoung Chul Ko. Driver's facial expression recognition in real-time for safe driving. *Sensors*, 2018.

[10] Dr. Maya Ingle Mrs Ayesha Butalia and Dr. Parag Kulkarni. Facial expression recognition for security. *International Journal of Modern Engineering Research*, 2, 2012.

[11] Dr Bageshree Pathak and Shriyanti Kulkarni. Speaker recognition system for home security using raspberry pi and python. *International Journal of Engineering and Technology*, 2018.

[12] Dr. Maya Ingle Mrs Ayesha Butalia and Dr. Parag Kulkarni. Facial expression recognition for security. *International Journal of Modern Engineering Research*, 2012.

[13] Rami Alazrai; C. S. George Lee. Real-time emotion identification for socially intelligent robots. *Proceedings*, 2012.

[14] Mariusz Szwoch Wioleta Szwoch Michal R. Wrobel Agata Kolakowska, Agnieszka Landowska. Emotion recognition and its application in software engineering. *6th International Conference on Human System Interactions (HSI)*, 2013.

[15] Lena Norberg Thomas Westin Peter Mozelius Mats Wiklund, William Rudenmalm. Evaluating educational games using facial expression recognition software-measurement of gaming emotion. *9th European Conference on Games Based Learning*, 2015.

[16] Triantafyllidis Georgios Christinaki Eirini, Vidakis Nikolas. A novel educational game for teaching emotion identification skills to preschoolers with autism diagnosis. *Computer Science and Information Systems*, 2014.

[17] Ali Khan Irfan, Azam Sajid. Feature extraction trends for intelligent facial expression recognition. *Informatica*, 2018.

[18] Paini Fabio. Deep learning for real-time emotion recognition from face images. *Master Thesis Project, Politecnico of Milan*, 2019.

[19] The front page of ai. `https://deepai.org`. DeepAI.

[20] Pierpaolo Ippolito. Feature extraction techniques. `https://towardsdatascience.com/feature-extraction-techniques-d619b56e31be`, 2019. Towards Data Science.

[21] Aayushi Bansal. A study of factors affecting face recognition. *International Journal of Advanced in Management, Technology and Engineering Sciences*, 2017.

[22] Pawel Tarnowski; Marcin Kolodziej; Andrzej Majkowskiy; Remigiusz Rak. Emotion recognition using facial expressions. *Procedia Computer Science*, 2017.

[23] Thomas S. Huang Pooya Khorrami, Tom Le Paine. Do deep neural networks learn facial action units when doing expression recognition? *ICCV*, IEEE.

[24] Mickael L.DerochecBrooke A.Burianeka Charles J.Limb Alison P.Goren Aditya M.Kulkarnia Julie A.Christensena Monita Chatte, Danielle J.Zion. Voice emotion recognition by cochlear-implanted children and their normally-hearing peers. *Hearing Research*, 2015.

[25] Xiaofan Lin John Burns Sherif Yacoub, Steve Simske. Recognition of emotions in interactive voice response systems. *8th European Conference on Speech Communication and Technology, Geneve*, 2003.

[26] Shaogang Gong Caifeng Shan and Peter W. McOwan. Facial expression recognition based on local binary patterns: A comprehensive study. *Image and Vision Computing*, 2009.

[27] D. Chan A. Mollahosseini and M. H. Mahoor. Going deeper in facial expression recognition using deep neural networks. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2016.

[28] Z. Meng P. Liu, S. Han and Y. Tong. Facial expression recognition via a boosted deep belief network. *IEEE Conf. Computer Vision and Pattern Recognition*, June 2014.

[29] Luoqi Liu Teng Li Yugang Han Nuno Vasconcelos Shuicheng Yan Xiangyun Zhao, Xiaodan Liang. Peakpiloted deep network for facial expression recognition. *Computer Vision ECCV*, 2016.

[30] J. Yim S. Park J. Kim H. Jung, S. Lee. Joint fine tuning in deep neural networks for facial expression recognition. *IEEE Int. Conf. Computer*, 2015.

[31] W. Lan H. Fu S. He, S. Wang and Q. Ji. Facial expression recognition using deep boltzmann machine from thermal infrared images. *Affective Computing and Intelligent Interaction*, 2013.

[32] Ying Chen Zhihao Zhang Zhan Wu, Tong Chen and Guangyuan Liu. Nirexpnet. Three-stream 3d convolutional neural network for near infrared facial expression. *Applied Sciences*, 2017.

[33] Shan Li and Weihong Deng. Deep facial expression recognition: A survey. *IEEE Transactions on Affective Computing*, 2020.

[34] Earnest Paul Ijjina and C. Krishna Mohan. Facial expression recognition using kinect depth sensor and convolutional neural networks. *13th International Conference on Machine Learning and Applications*, 2014.

[35] Kwang-Eun KoKwee-Bo Sim. Emotion recognition in facial image sequences using a combination of aam with facs and dbn. *International Conference on Intelligent Robotics and Applications*, 2010.

[36] M. H. Mahoor A. Mollahosseini, B. Hasani. Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, 2018.

[37] Z. Meng; P. Liu; S. Han and Y. Tong. Facial expression recognition via a boosted deep belief network. *Computer Vision and Pattern Recognition*, 2014.

[38] D. Chan A. Mollahosseini and M. H. Mahoor. Going deeper in facial expression recognition using deep neural networks. *IEEE Winter Conference on Applications of Computer Vision*, 2016.

[39] Baoye Song Weibo Liu Yurong Li Abdullah M Dobaie Nianyin Zeng, Hong Zhang. Facial expression recognition via learning deep sparse autoencoders. *Neurocomputing*, 2018.

[40] Alberto F De Souza Thiago OliveiraSantos AndrÃ© Teixeira Lopes, Edilson de Aguiar. Facial expression recognition with convolutional neural networks: coping with few data and the training sample order. *Pattern recognition*, 2017.

[41] Yijun Gan. Facial expression recognition using convolutional neural network. *2Nd International Conference on Vision, Image and Signal Processing*, 2018.

[42] Ping Hu Shandong Wang Liang Sha Yurong Chen. Holonet Anbang Yao, Dongqi Cai. Towards robust emotion recognition in the wild. *18th ACM International Conference on Multimodal Interaction*, 2016.

[43] Shandong Wang Anbang Yao Ping Hu, Dongqi Cai and Yurong Chen. Learning supervised scoring ensemble for emotion recognition in the wild. *19th ACM International Conference on Multimodal Interaction*, ICMI.

[44] M. Kim M. Shin and D. Kwon. Baseline cnn structure analysis for facial expression recognition. *25th IEEE Int. Symp. Robot and Human Interactive Communication*, 2016.

[45] Cristian Canton Ferrer Sarah Adel Bargal, Emad Barsoum and Cha Zhang. Emotion recognition in the wild from videos using images. *18th ACM International Conference on Multimodal Interaction*, 2016.

[46] Peter W. McOwan Caifeng Shan, Shaogang Gong. Facial expression recognition based on local binary patterns: A comprehensive study. *Image and Vision Computing*, 2009.

[47] Xiangyun Zhao; Xiaodan Liang; Luoqi Liu; Teng Li; Yugang Han; Nuno Vasconcelos; Shuicheng Yan. Peak-piloted deep network for facial expression recognition. *Computer Vision ECCV*, 2016.

[48] Yong Du Kaihao Zhang, Yongzhen Huang and Liang Wang. Facial expression recognition based on deep evolutional spatial-temporal networks. *IEEE Transactions on Image Processing*, 2017.

[49] B. Hasani A. Mollahosseini and M. H. Mahoor. Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, 2018.

[50] A. Dapogny and K. Bailly. Investigating deep neural forests for facial expression recognition. *13th IEEE Int. Conf. Automatic Face Gesture Recognition*, IEEE.

[51] Dian Tjondronegoro Ligang Zhang, Brijesh Verma and Vinod Chandran. Facial expression analysis under partial occlusion: A survey. *ACM Comput.*, 2018.

[52] Radhika M Pai Veena Mayya and MM Manohara Pai. Automatic facial expres- sion recognition using dcnn. *Procedia Computer Science*, 2016.

[53] S. Shan M. Liu, S. Li and X. Chen. Auaware deep networks for facial expression recognition. *10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, 2013.

[54] Kuniaki Noda; Hiroaki Arie; Yuki Suga; Tetsuya Ogata. Multimodal integration learning of robot behavior using deep neural networks. *Robotics and Autonomous Systems*, 2014.

[55] Dewan Md. Farid Muhammad Akram Hossain Li Zhang, Ming Jiang. Intelligent facial emotion recognition and semantic-based topic detection for a humanoid robot. *Expert Systems with Applications*, 2013.

[56] Nicolas Hamelin, Othmane El Moujahid, and Park Thaichon. Emotion and advertising effectiveness: A novel facial expression analysis approach. *Journal of Retailing and Consumer Services*, 2017.

[57] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, 2001.

[58] Q. Ruan R. Zhi, M. Flierl and W. B. Kleijn. Graphpreserving sparse nonnegative matrix factorization with application to facial expression recognition. part b. *IEEE Transactions on Systems, Man, and Cybernetics,*, 2011.