POLITECNICO DI MILANO

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

Deep Learning for real-time emotion recognition from face images

Supervisor: Pietro Cerveri

Thesis of:
Fabio Paini
Student ID: 874550

Academic Year 2018-2019

Acknowledgments

I'd like to thank everybody who supported me in these years of university and especially in these last months of thesis work.

I thank the Politecnico di Milano and all its professors who helped me acquire the necessary knowledge and tools to complete this project.

I thank my friends who shared with me this time as students at Politecnico, because they made this time more valuable than any courses could.

I thank my friends who did not attend Politecnico, because they always remained my friends even though I was little more than a ghost appearing every now and then.

I thank my thesis supervisor, Professor Pietro Cerveri, for the chance to work on such an interesting topic and follow me throughout this thesis work, not only from a didactic perspective but always remembering that we are humans before engineers.

I thank the Seminario Vescovile di Parma, which welcomed me for a little part of my career and reminded me of what's really important in life.

I thank my family for supporting me in my whole life and for accepting that, especially lately, even the few times that I was at home I would constantly think of working. I'll try to make up for that.

I especially want to thank my girlfriend for always being at my side and for accepting me at hers. Because of you testing this project was really hard because every time you appeared in my mind the only expression I could show was a smile.

Sommario

Il riconoscimento di espressioni facciali (FER, dall'inglese *Facial Expression Recognition*) è un compito che sta diventando sempre più rilevante in molti campi, dalla social robotics alla sanità. In questo contesto, tecniche di deep learning hanno recentemente mostrato risultati promettenti. Tuttavia, questo problema è sempre stato approcciato come qualsiasi altro problema di classificazione senza considerare le peculiarità del FER, in particolare il fatto che il modo in cui un'espressione è interpretata è solitamente soggettivo e non sempre molto chiaro. In questo lavoro, delle tecniche di deep learning classiche (in particolare reti neurali convoluzionali) sono usate per affrontare il problema del riconoscimento di espressioni facciali in tempo reale (sviluppando un'apposita applicazione che riconosca le emozioni da immagini ottenute attraverso una webcam) e i risultati sono analizzati per capire se questo metodo tiene conto delle caratteristiche specifiche del FER o come questo potrebbe essere fatto. A questo fine, un piccolo dataset è stato creato facendo annotare le stesse immagini a più persone, in modo da avere annotazioni che non forzano una singola classe per ogni immagine ma considerano la possibilità che un'immagine appartenga (in misura diversa) a varie classi insieme.

I risultati mostrano che i classificatori sviluppati imparano alcune caratteristiche particolari di questo compito (ovvero la differenza in riconoscibilità fra varie emozioni e l'ambiguità di alcune espressioni facciali) e possono ottenere risultati migliori degli umani quando sono usate etichette singole. È tuttavia necessario esplorare altri approcci che tengano conto del fatto che spesso non c'è una singola verità oggettiva quando si tratta di FER, in modo da sviluppare classificatori che effettivamente interpretino le espressioni facciali come farebbe un umano.

Introduzione

Le relazioni con le altre persone sono fondamentali nella vita di ogni persona e la comunicazione verbale non è più importante di tutti quei sottili movimenti con cui i nostri corpi parlano. Il nostro corpo può dire molto riguardo il nostro stato mentale e

ci permette di esprimerci oltre quello che le parole possono dire. Una ridotta capacità di esprimere e riconoscere le emozioni, essendo una funzione così fondamentale per gli uomini, può essere un segno di malattie neuropsichiatriche o la conseguenza di traumi. D'altra parte, osservare dei miglioramenti può indicare guarigione. Uno dei modi più importanti con cui le persone si esprimono e si connettono agli altri è attraverso le emozioni e particolarmente come sono espresse nei nostri volti. Il riconoscimento di espressioni facciali (FER) ha conosciuto una recente popolarità e le più moderne tecniche sono usate per permettere alle macchine di eseguire automaticamente questo compito. Il FER è rilevante in vari ambiti, dalle applicazioni diagnostiche alla robotica affettiva.

Il riconoscimento di espressioni facciali dalle immagini è particolarmente importante perché non richiede attrezzatura complessa e permette di acquisire i dati con una camera senza recare disturbo ai soggetti. Recentemente, l'aumento dei dati disponibili e del potere computazionale ha portato all'uso di tecniche di deep learning, imponendo un nuovo stato dell'arte. Un fatto che non è solitamente considerato è che lo stesso volto può essere interpretato come se esprimesse emozioni diverse, con risposte diverse sia da persone diverse che dalla stessa persona che lo vede in momenti diversi. C'è quindi bisogno di capire se i classificatori allenati con metodi tradizionali sono in grado di prendere in considerazione le peculiarità del FER (ad esempio espressioni facciali ambigue o che mostrano più emozioni allo stesso tempo) e come migliorare questi metodi in modo da approcciare questo problema nel modo giusto.

Materiali e metodi

Lo scopo del progetto era di sfruttare tecniche di deep learning (in particolare usando reti neurali artificiali) per sviluppare un classificatore per il FER che potesse funzionare in tempo reale, capire se questo è un buon approccio al riconoscimento di espressioni facciali e trovare quali informazioni (se ce ne sono) potrebbero essere usate per migliorare la tecnica per questo specifico compito.

I vari dataset disponibili per il FER sono stati studiati per trovare quelli più adatti. Alla fine, i dataset AffectNet e CK+ sono stati scelti. Il primo contiene immagini che sono state raccolte da varie fonti e che presentano differenze nelle dimensioni, illuminazione, sesso, età e gruppo etnico dei soggetti, qualità, orientamento del volto e "livello di spontaneità" (alcune immagini mostrano chiaramente espressioni finte, mentre altre contengono foto di emozioni genuine). Queste caratteristiche rendono questo dataset perfetto per una rete neurale che dovrebbe lavorare con immagini prese da webcam in condizioni enormemente differenti l'una dall'altra e l'alto numero di

esempi è adatto per il Deep Learning. Le immagini di CK+, invece, mostrano soggetti di fronte con una buona illuminazione e le espressioni mostrate seguono fedelmente le regole descritte nella FACS Investigators Guide. A causa di questo e del fatto che è il dataset più usato in questo campo, è stato scelto di usare CK+ per validare ulteriormente i classificatori allenati e per verificare come si comportano con compiti più "facili". Per entrambi i dataset, le classi usate erano *rabbia*, *biasimo*, *disgusto*, *paura*, *felicità*, *neutrale*, *tristezza* e *sorpresa*.

AffectNet contiene volti a varie scale, pose e posizioni all'interno delle immagini (esattamente come succederebbe con immagini ottenute da una webcam). La rete neurale sprecherebbe quindi molte risorse per tenere conto di tutte le possibili posizioni del volto nelle immagini. Per evitare questo, le immagini in ingresso (sia nel dataset che durante l'uso reale della rete) sono preprocessate da un algoritmo di normalizzazione che allinea i volti rilevati con un template, rimuovendo allo stesso tempo informazioni inutili come capelli e sfondo. Questo algoritmo elabora l'immagine con i seguenti passi (dal punto 2 in avanti, le operazioni sono eseguite su ogni volto separatamente):

1. tutti i volti nell'immagine sono rilevati e ritagliati
2. 68 riferimenti facciali sono localizzati usando un altro rilevatore pre-allenato
3. 3 punti, corrispondenti ai due occhi e la bocca, sono calcolati come combinazione lineare delle posizioni dei riferimenti
4. l'immagine è trasformata attraverso una trasformazione affine che posiziona i 3 punti nelle posizioni corrispondenti del template (allo stesso tempo, l'immagine è anche ridimensionata alla dimensione di input della rete)
5. opzionalmente, l'immagine è convertita in scala di grigio
6. i colori dell'immagine sono normalizzati in modo da avere media e deviazione standard fissate

L'architettura di rete è stata scelta come una serie di strati di convoluzione-max pooling-ReLU seguiti da due strati fully connected (con l'ultimo che effettua la vera classificazione). Da questo schema di base, varie reti sono state create modificando i parametri. La maggior parte sono state allenate su AffectNet mentre poche altre su CK+.

Un'applicazione Python è stata creata per usare le reti neurali su immagini acquisite da webcam, permettendo così la visione delle predizioni dei classificatori in tempo reale.

Separatamente da tutto questo, è stata considerata la possibilità di usare annotatori multipli per avere più dati riguardanti l'espressione delle emozioni per ogni immagine. Un dataset di questo tipo (WebAN) è stato costruito creando un sito web dove le persone potessero ri-etichettare 550 immagini estratte da AffectNet. Hanno contribuito

89 annotatori con 7466 annotazioni, 13-14 per immagine. Le etichette risultanti erano alquanto diverse da quelle originali, ma soprattutto contenevano informazioni "probabilistiche" riguardo le classi anziché considerarle completamente distinte. Viene suggerito che questi nuovi dati potrebbero essere usati per allenare reti neurali che tengono conto dell'ambiguità della maggior parte delle immagini. Questo potrebbe essere fatto o considerando il compito come un problema di regressione anziché di classificazione, oppure usando funzioni di errore adatte a etichette sfocate. In particolare, due funzioni di errore (chiamate Entropia Incrociata Categorica Sfocata e Entropia Incrociata Categorica Sfocata ad Etichetta Singola) sono qui proposte:

$$\mathcal{F} = -2\log\left(\sum_{c=1}^{C}\sqrt{t_c p_c} + \epsilon\right) \quad \mathcal{F}^* = -2\log\left(\sum_{c=1}^{C}\sqrt{t_c^* p_c}\right) + \alpha\left(1 - \sum_{c=1}^{C} p_c^2\right)$$

dove t_c e p_c sono gli elementi dei vettori, rispettivamente, obiettivo e predizione relativi a una particolare classe c.

Tutte le reti neurali sono state poi validate con AffectNet, CK+ e WebAN (sia con le etichette originali che con quelle acquisite dal sito web). Le annotazioni raccolte sono anche state analizzate per vedere se gli annotatori fossero in grado di accordarsi su una sola etichetta per ogni immagine, se le reti non riuscissero a dare risposte ben definite per le stesse immagini che erano considerate ambigue dagli umani e per avere una accuratezza di riferimento relativa a un "classificatore umano".

Risultati

Table 1: *Miglior accuratezza raggiunta dalle varie reti*

Dataset di allenamento	Dataset di validazione	Accuratezza massima
AffectNet	AffectNet	48.7%
	CK+	74.3%
	WebAN (etichette di AffectNet)	77.4%
	WebAN (etichette dal sito)	46.5%
CK+	AffectNet	23.3%
	CK+	98.1%
	WebAN (etichette di AffectNet)	24.9%

La Tabella 1 mostra la massima accuratezza raggiunta collettivamente sui vari dataset dalle reti allenate su AffectNet e CK+. I test sulle metriche calcolate sulle singole classi hanno mostrato la presenza di una differenza in valore significativa fra le varie classi, con metriche particolarmente alte per la classe *felicità* e particolarmente basse per la classe *neutrale*.

L'analisi delle annotazioni ha indicato un basso accordo tra gli annotatori sulla maggior parte delle immagini, a parte rare eccezioni. È stata trovata una correlazione significativa fra il disaccordo degli annotatori sull'etichetta di un'immagine e predizioni "incerte" delle reti (intese come quelle per cui l'output relativo alle varie classi è simile anziché essere molto più alto per una sola rispetto alle altre). Quando gli annotatori sono stati valutati rispetto alle etichette contenute in AffectNet, hanno raggiunto un'accuratezza media del 42.9%, aumentando a 46.7% quando tutti gli annotatori sono stati considerati insieme usando un sistema di voto maggioritario.

Discussione e conclusioni

Le reti non hanno raggiunto risultati comparabili con lo stato dell'arte, ma va notato che le migliori reti in letteratura sono molto più profonde di quelle considerate in questo lavoro a causa delle limitate risorse a disposizione. È interessante notare che la rete con accuratezza migliore non è stata una delle più complesse che sono state allenate, mostrando così che una rete più grande non porta sempre a risultati migliori ed è più importante calibrare bene i parametri. L'accuratezza ottenuta appare molto più rispettabile quando i risultati delle reti e di veri uomini sono confrontati nello stesso dataset. L'umano medio, infatti, ha un'accuratezza ancora più bassa delle reti neurali. Le metriche riguardanti l'accordo mostrano come questo non è dovuto al fatto che le etichette a disposizione erano semplicemente sbagliate, ma al fatto che le immagini mostravano espressioni ambigue, cosicché non fosse possibile dare una risposta definita e i vari annotatori non potessero accordarsi sulla stessa etichetta. La correlazione tra l'"accordo" delle predizioni e quello degli annotatori mostra come le reti neurali hanno implicitamente imparato quali immagini sono più ambigue, anche senza essere state addestrate per farlo e nessun dato a tal riguardo è stato loro fornito. I valori di accuratezza mostrano che allenare una rete su un dataset "difficile" come AffectNet permette la generalizzazione anche ad altri dataset, mentre usare un dataset semplice come CK+ durante l'allenamento non è adatto per imarare espressioni più generali. I risultati hanno anche dimostrato il fatto intuitivo che alcune emozioni sono più facili da riconoscere di altre.

I dati raccolti suggeriscono che probabilmente i metodi di classificazione tradizionali non sono perfettamente adatti per il FER e un approccio più probabilistico all'annotazione dovrebbe essere usato per tener conto della natura spesso ambigua delle espressioni facciali. Anche la differenza in complessità nel riconoscere le varie emozioni dovrebbe essere considerata quando si agisce in base all'espressione rilevata: un volto felice può essere riconosciuto relativamente facilmente, ma è facile confondere un volto

che mostra biasimo con qualcos'altro.

Abstract

Facial expression recognition (FER) is a task that is becoming ever more relevant in many fields, ranging from social robotics to healthcare. In this context, deep learning techniques have shown promising results recently. However, this problem has always been approached as any other classification task without considering the peculiarities of FER, in particular the fact that how an expression is interpreted is usually subjective and not always very clear. In this work, classical deep learning techniques (in particular convolutional neural networks) are used to confront with the task of real-time facial expression recognition (developing an application that recognizes emotions from webcam images) and the results are analyzed in order to understand if this approach takes into account the specific features of FER or how this could be accomplished. To this end, a small dataset has been created by making multiple people label the same images, in order to have annotations that do not force single classes on each image but consider the possibility of an image belonging (in different measures) to various classes at the same time.

The results show that the developed classifiers do learn some particular characteristics of this task (i.e. difference in recognizability among the various emotions and ambiguousness of some facial expressions) and can perform better than humans when single labels are forced. It is however necessary to explore other approaches that take into account the fact that often there is not a single ground truth when talking about FER, in order to develop classifiers that actually interpret facial expressions like humans do.

Introduction

Relations with other people is vital in the life of every person and verbal communication is not more important than all those subtle movements by which our bodies talk. Our body can tell a lot about our mental state and make it possible for us to express ourselves beyond what words can say. A reduced ability to express and recognize emotions, being such a fundamental function for humans, can be a sign of neuropsychiatric

diseases or the consequence of injuries. On the other hand, observing improvements in it can indicate healing. One of the most important ways by which people express themselves and connect to each other is through emotions and particularly how they are expressed in our faces. Facial expression recognition (FER) has known a recent popularity and the most modern approaches are used to allow machines to automatically perform this task. FER is relevant in a range of different fields, from diagnostic applications to affective robotics.

Facial expression recognition from images is particularly important because it does not require advanced equipment and allows to acquire the data easily with a camera without inconveniencing the subjects. Recently, the increase in available data and computing power has led to the use of deep learning techniques, reaching state-of-the-art results. A fact that is not usually addressed is that the same face could be recognized as expressing different emotions, with different answer from both different people and the same person seeing the same face twice. There is thus a need to understand if classifiers trained with traditional methods can address the peculiarities of FER (e.g. ambiguous facial expressions or ones that show multiple emotions at the same time) and how to improve these methods in order to properly approach this task.

Materials and methods

The goal of the project was to exploit deep learning techniques (in particular using artificial neural networks) to develop a classifier for FER that could be run in real time, investigate whether or not this is a good approach for facial expression recognition and find which information (if any) could be used to improve the technique for this specific task.

The various available datasets for FER were investigated in order to find the most suitable ones. In the end, the AffectNet and the CK+ datasets were selected. The first one contains images that were gathered from various sources, having differing sizes, light conditions, sex, age and ethnic group of the subjects, image qualities, face orientations and "level of spontaneousness" (some images are clearly posed, while other show candid shots of genuine emotions). These characteristics make this dataset the perfect fit for a neural network that should work with webcam images taken in wildly different conditions and the great number of examples is also suitable for Deep Learning purposes. The CK+ images, on the other hand, show the subjects from the front with a good illumination and the shown expressions closely follow the stereotypical rules described in the FACS Investigators Guide. Because of this and the fact that it is the most standard dataset in this field, it was chosen to use the CK+ dataset for

further validation of the trained classifiers and to check how they perform on "easier" tasks. For both datasets, the used classes were *anger*, *contempt*, *disgust*, *fear*, *happiness*, *neutral*, *sadness* and *surprise*.

The AffectNet dataset contains faces at various scales, orientations and positions inside the images (just like it would happen with images taken from a webcam). The neural network would hence need to waste a lot of computing resources to account for all the possible face localization differences in the input images. To avoid this, the input images (both in the dataset and during actual use of the network) are preprocessed by a normalization algorithm that aligns the detected faces to a standard template, at the same time removing useless information like hairs and background. This algorithm processes the image in the following steps (from step 2 onward, the operations are performed on each detected face separately):

1. all faces in the image are automatically detected and cropped to their bounding box
2. 68 facial landmarks are located using another pre-trained detector
3. 3 markers, corresponding to the two eyes and the mouth, are computed as linear combinations of the positions of the landmarks
4. the image is morphed through an affine transformation that places the 3 markers in the corresponding positions of the template (at the same time, the image is also resized to the input shape of the network)
5. optionally, the image is converted to grayscale
6. the image colors are normalized to have a fixed mean and standard deviation

The network architecture was selected as a series of convolution-max pooling-ReLU layers followed by two fully connected layers (the last one being the one actually performing the classification). From this basic template, various networks were created by tweaking the parameters. Most of them were trained on the AffectNet dataset while a few other on the CK+ dataset.

A Python application was created to use the neural networks on images acquired by a webcam, allowing to see the predictions of the classifiers in real time.

Separately from all of this, the possibility of using multiple annotators to have more data about the emotion expression of each image was considered. A dataset of this kind (called WebAN) was assembled by creating a website where people could relabel 550 images extracted from the AffectNet dataset. 89 annotators contributed with 7466 annotations, 13-14 for each image. The resulting labels were quite different from the

original ones, but most importantly they contained "probabilistic" information about the classes rather than considering them completely distinct. It is suggested that these new data could be used to train neural networks which actually take into account the ambiguity of most images. This could be done either considering the task a regression problem rather than a classification one, or by using custom loss functions suitable for fuzzy labels. In particular, two loss functions (called Fuzzy Categorical Cross Entropy and Single-label Fuzzy Categorical Cross Entropy) are here proposed:

$$\mathcal{F} = -2\log\left(\sum_{c=1}^{C}\sqrt{t_c p_c} + \epsilon\right) \quad \mathcal{F}^* = -2\log\left(\sum_{c=1}^{C}\sqrt{t_c^* p_c}\right) + \alpha\left(1 - \sum_{c=1}^{C} p_c^2\right)$$

where t_c and p_c are the elements of respectively the target and prediction vectors relative to a particular class c.

All the neural networks were then validated on the AffectNet, CK+ and WebAN (both with the original labels and the ones acquired from the website) datasets. The gathered annotations were also analyzed to see if the annotators could agree on a single label for each image, whether or not the networks could not give a clear-cut answer for the same images that were considered ambiguous by humans and to have a human reference accuracy score.

Results

Table 2: *Best accuracy scores reached by the various networks*

Training set	Validation set	Max. accuracy
AffectNet	AffectNet	48.7%
	CK+	74.3%
	WebAN (AffectNet labels)	77.4%
	WebAN (website labels)	46.5%
CK+	AffectNet	23.3%
	CK+	98.1%
	WebAN (AffectNet labels)	24.9%

Table 2 shows the maximum accuracy scores collectively reached on the various datasets by the networks trained on the AffectNet and on the CK+ datasets. Tests on the per-class metrics showed that there was a significant difference in values among the various classes, with particularly high metrics for the *happiness* class and particularly low ones for the *neutral* class.

The analysis of the annotations indicated a low agreement between the annotators on most images, except for some rare exceptions. There was a significant correlation

between disagreement of the annotators on the label of an image and "uncertain" predictions by the networks (in the sense that the output for the various classes was similar rather than one class having an output much higher than the others). When the annotators were evaluated against the AffectNet labels, they reached an average accuracy of 42.9%, raising to 46.7% when all annotators were considered together by using a majority vote system.

Discussion and conclusions

The networks did not reach state-of-the-art performance, but it must be noted that the best-performing networks in literature are much deeper than the ones considered in this work due to resource limits. Interestingly, the best-performing network was not one of the most complex ones that were trained, showing that a bigger network not always gives better results and it is more important to properly tune the parameters. The obtained accuracy score becomes much more respectable when the performance of the networks and of real humans is compared on the same dataset. The average human, in fact, has an even lower accuracy than the neural networks. The agreement metrics show how this is not due to the fact that the available labels were simply wrong, instead most images showed ambiguous emotion expressions, so that a definitive answer could not be given and the various annotators could not agree on the same label. The correlation between the "agreement" of the predictions and that of the annotators show how the neural networks implicitly also learned which images are more ambiguous, even if they were never instructed to do so and no related data was given them. The accuracy scores show that training on a "hard" dataset like AffectNet allows for a generalization on other datasets too, while using a simple dataset like CK+ during training is not suitable for learning more general expressions. The results also proved the intuitive fact that some emotions are easier to recognize than others.

The acquired data suggest that traditional classifications approaches are probably not perfectly suited for FER and a more probabilistic approach to labeling should be taken in order to account for the often ambiguous nature of facial expressions. The difference in complexity of recognizing the various emotion should also be taken into account when the detected expression is acted upon: an happy face can be detected with relative ease, but it is easy to mistake a face showing contempt for something else.

Contents

List of Figures

List of Tables

List of symbols and abbreviations

API	Application Programming Interface
CMR	Convolution-Maxpooling-ReLU block
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DAE	Deep Autoencoder
DBN	Deep Belief Network
DSP	Digital Signal Processor
FACS	Facial Action Coding System
FER	Facial Expression Recognition
FPGA	Field Programmable Gate Array
GPU	Graphical Processing Unit
GUI	Graphical User Interface
LBP	Local Binary Pattern
MSE	Mean of Squared Errors
NE	Network Ensemble
NF	Deep Neural Forest
OS	Operating System
ReLU	Rectified Linear Unit

RNN	Recurring Neural Network
SIFT	Scale-Invariant Feature Transform
SVM	Support Vector Machine

Chapter 1

Introduction

1.1 Context

Φύσει μεν εστιν άνθρωπος
ζῶν πολιτικών
— Aristotle, 4th century BC

"Man is by nature a social animal", so Aristotle wrote more than 2000 years ago. Relations with other people is vital in the life of every person and verbal communication is not more important than all those subtle movements by which our bodies talk. Our body can tell a lot about our mental state and make it possible for us to express ourselves beyond what words can say.

A reduced ability to express and recognize emotions, being such a fundamental function for humans, can be a sign of neuropsychiatric diseases (Alzheimer [1], Parkinson [2], depression [3], autism [4], schizophrenia and bipolar disorder [5], etc.) or the consequence of injuries [6] [7]. On the other hand, observing improvements in it can indicate healing. Currently, these observations are usually done by specialized medical operators who:

1. are few with respect to the number of patients and this can be hardly changed due to the cost to pay such an expert

2. even when using numerical scales, can only make subjective considerations

An automatic method to assess this ability would make it much more cost effective, allow for more patients to be monitored without increasing the number of operators and make this ability quantifiable in a consistent way, helping in functional evaluation targeted at studying the best treatment for each person.

In recent years, a common trend has emerged of considering machines not simply tools to be used by humans, but smart (or apparently smart) helpers which collaborate with them towards a common goal. Affective computing and social robotics are two branches of computer science which aim at making artificial "objects" behave with people in such a way to make them "connect", communicating not only with sterile text-based or similar interfaces: human interactions are much more than words and this is critical to be understood in order to properly communicate with people, in both directions.

Neuromarketing is another field which recognizes the importance of understanding the internal mental state of people when interacting with them [8].

One of the most important ways by which people express themselves and connect to each other is through emotions and particularly how they are expressed in our faces [9]. Facial expression recognition (FER) has known a recent popularity, as shown in Figure 1.1, and the most modern approaches are used to allow machines to automatically perform this task.

The applications of FER are countless. Just as few examples:

- diagnosis [10] and study [11] of autism
- driver fatigue detection [12]
- security applications [13]
- emotion detection system used by the disabled to assist a caretaker [14]
- socially intelligent robots [15]

1.2 Work motivation

As explained, emotion recognition is becoming more and more relevant and an enabling technology in various fields. Facial expression recognition from images is particularly important because it does not require advanced equipment and allows to acquire the data easily with a camera without inconveniencing the subjects.

This task was first approached by creating automatic classifiers that were however trained on engineered features such as local binary patterns [16][17] and non-negative matrix factorization [18].

More recently, the increase in available data and computing power has led to the use of deep learning techniques [19], reaching state-of-the-art results. This however means that the feature extraction is made by black-boxes and it is thus hard to understand if the classifiers are actually learning how to recognize emotions as a human would or

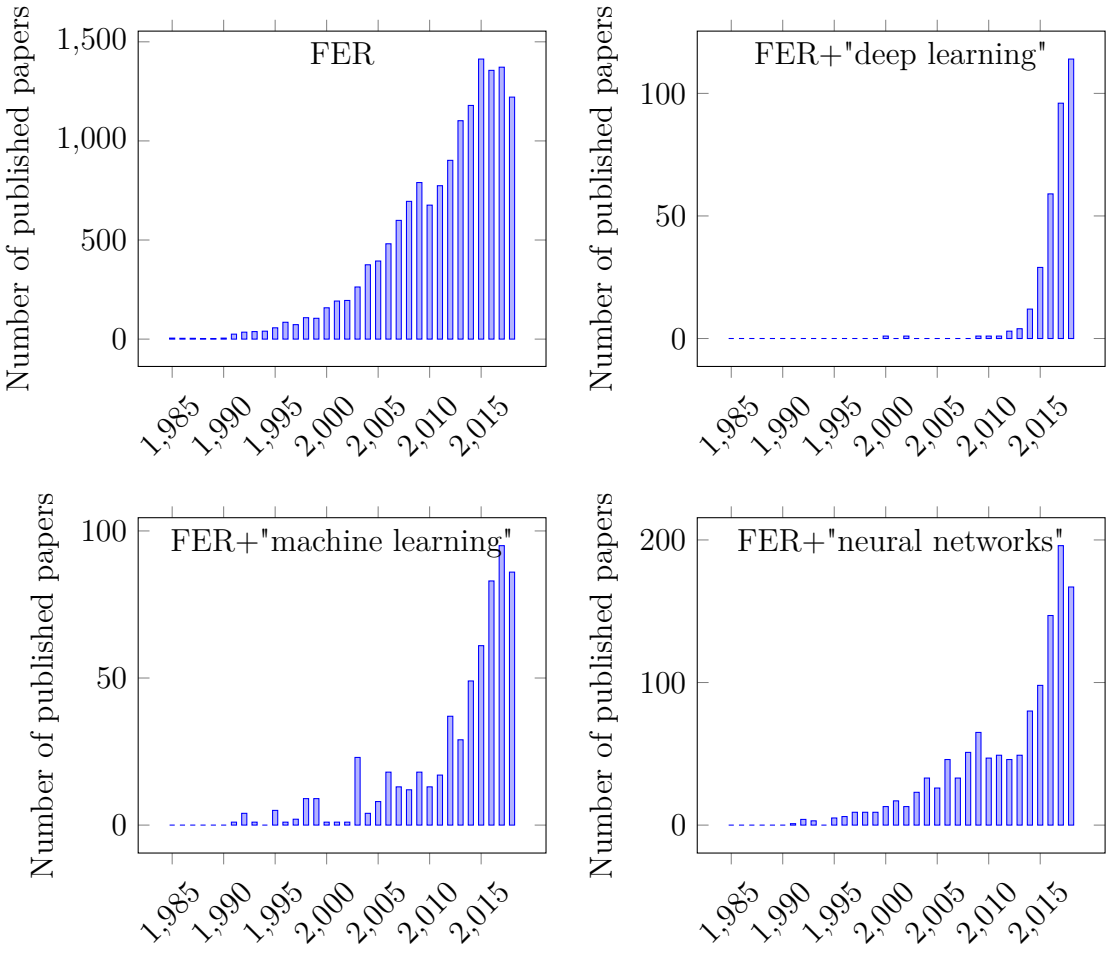


Figure 1.1: *Number of published papers in the years from 1985 to 2018 in various mainstream journals, divided by topic. The data was obtained by querying the Web of Science platform (`https://www.webofknowledge.com`) with the keywords written inside the charts;* FER *stands for both "facial expression recognition" and "facial emotion recognition".*

if they categorize the images without also learning the related information that people associate them with.

A fact that is not usually addressed is that FER is not a classification task like any other since there is not a clear distinction of the classes. While classical classification is based on the assumption that there is a *correct answer*, in FER it is often more important how people interpret them. Humans are not deterministic machines and the same face could be recognized as expressing different emotions, with different answer from both different people and the same person seeing the same face twice. Additionally, sometimes the same facial expression can show multiple emotions at the same time. Figure 1.2 shows some images containing facial expressions, with some showing clear emotions while other are more ambiguous.

There is thus a need to understand if classifiers trained with traditional methods

Figure 1.2: *Examples of faces showing various facial expressions*

can address the peculiarities of FER (e.g. ambiguous facial expressions or ones that show multiple emotions at the same time) and how to improve these methods in order to properly approach this task. A good classifier should be able to recognize expressions not only as distinct categories of emotions but also as combinations of them, all the while being able to generalize for images showing different poses, illumination, subject ethnicity, etc.

1.2.1 Proposed work

The goal of this project is firstly to develop a basic framework for designing, training and testing classifiers that could be useful for FER done on real-time webcam images, studying the best ways to approach the problem. The images could either be considered singularly or together as a video sequence. In this project, still images are used since they have a larger number of applications (a video can be converted to a sequence of still images, but not vice versa).

A fundamental step is the choice of the datasets to be used for training and validation. There are various datasets containing still images of faces with a corresponding expression label with differing characteristics. A good dataset should have a number of examples sufficient for deep learning applications and contain pictures taken in wildly different conditions in order to allow the networks to properly generalize. A comparison between the results obtained from datasets that are more general and ones where

there are more "artificial" conditions should also be performed in order to understand the limits of using a dataset that does not properly represent the variability of real-life situations.

While the datasets contain the necessary data, a preprocessing step could be needed before being fed to the training algorithm. A good way to provide the classifiers with useful inputs (which could reduce the computational power wasted on useless information) should be investigated.

Neural networks are chosen as the main technology for the classifiers, considering the outstanding results that they have recently shown in image classification tasks in general and FER in particular. The networks can not be too complex due to hardware and time limitations, but should try to fully exploit the techniques that are most popular in the state of the art. Various different tests need to be performed, exploring the possible architectures and parameters, in order to both find the best possible networks for this task and finding information that are not limited to specific implementations but are more linked to the classification itself.

Since it is often useful to get information about the emotions of a person as soon as they are expressed, the applicability of these networks to real-life situations needs to be studied by developing a real-time application that uses them.

These networks should then be analyzed in order to evaluate their generalization power, assess the reliability of the results and, most importantly, understand if and which related features are learned by the classifiers while trying to categorize the emotion expressions.

A study also needs to be done regarding methods capable of accounting for the often overlooked peculiarities of FER, in particular the fact that in many cases a single label is not enough to represent a facial expression. For this, a dataset containing "probabilistic" labels is needed. No such dataset is currently available, so a new one must be created.

1.2.2 Expected results

The expectations are that:

1. the developed classifiers are able to perform adequate classifications on a very generalized range of images
2. a webcam-based desktop application that uses those classifiers to recognize facial expressions in real time and show them to a user/operator is developed
3. the developed classifiers perform better on "easy" images with good illumination, exaggerated expressions and frontal poses

4. classifiers trained on those same "easy" images are incapable of good generalization

5. some emotions are easier to recognize than others

6. some images show ambiguities when annotated by different humans

7. using more information than a single label improves the performance of the classifiers

1.3 State of the art

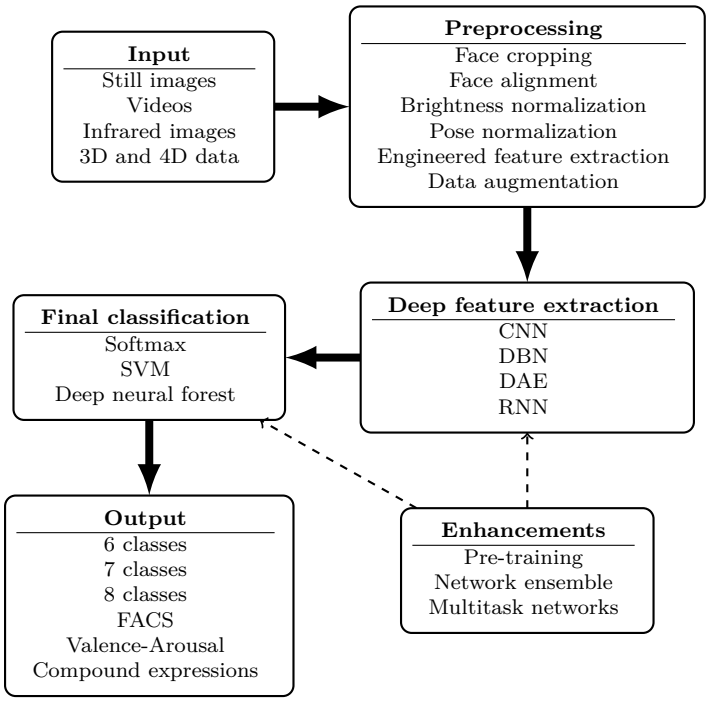


Figure 1.3: *Scheme of the basic processing pipeline found in the state of the art*

Recently, the research on FER has focused on deep learning methods to approach this classification task. Among all the various proposed methods, a common scheme is present:

1. preprocessing of the images (usually involving face detection)

2. automatic feature extraction through the deep neural networks

3. classification using the extracted features

The various methods then distinguish themselves on how they perform these phases and the data they use.

While the task of FER is by definition performed on visual information of faces, which exact data is used can vary. Most classifiers work on still RGB or grayscale images [16, 20, 21, 22], while others use videos [23, 24]. In order to overcome some of the limitations of normal images, such as illumination variations and pose-related distortions, there are cases of FER done on infrared images [25, 26] and 3D data [19, 27].

The preprocessing phase can vary, but 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. Virtually all of the proposed methods start the processing by detecting faces (most commonly using the Viola-Jones detector [28]) and cropping the image to them. The extracted face can then be subjected to other transformations:

- alignment using facial landmarks (usually eyes and mouth), detected through algorithms such as Active Appearance Models [29] and other deep neural networks [30]
- pose correction to make the images look like they were taken from the front [31, 32]
- brightness normalization [33, 34]
- dimensional reduction of the information by extracting features such as local binary patterns (LBP) [16] or Scale-invariant feature transform (SIFT) features [35]

Some authors [36, 37, 38], instead of trying to remove the unrelated data, model the classifier so that it would take into account some of the variability not directly linked to the emotion expression. An additional step that is necessary when small datasets are used is data augmentation, e.g. by applying random rotations, scaling and noise [39, 40]. A different approach for coping with small datasets is to pre-train the feature extraction stage on bigger datasets concerning other related tasks (in particular face recognition), sometimes with well-known networks developed for those other tasks [41], and then fine-tune it with the smaller facial expression dataset [42, 43].

The deep features extractor is always a neural network, but with differing topologies:

- convolutional neural networks (CNN) [21, 22, 40]
- deep belief networks (DBN) [20]
- deep autoencoders (DAE) [29]

- recurrent neural networks (RNN) [30]

Some methods join the feature extraction phase and the classification one, training them together [21, 22, 44] while others extract the features first and then train the actual classifier separately [45].

When the classification is done end-to-end without separating it from the feature extraction, it is most often performed using a softmax layer [21, 22, 29, 46]. Other approaches include the use of support vector machines (SVM) [16, 45, 47] and deep neural forests (NF) [44].

Instead of using a single network to classify all the images, some methods are based on the idea of combining the results or extracted features of various specialized networks in a network ensemble (NE) [30, 34].

As for how the result of the classification should be represented, not everyone agrees. Almost all methods classify emotions in a categorical model containing six basic expressions (*anger*, *disgust*, *fear*, *happiness*, *sadness* and *surprise*) [20, 21, 46], sometimes adding a *neutral* class [20, 29, 47] and, rarely, the *contempt* class is added to the basic expressions [22, 47]. A recent development is the introduction of a dimensional model (valence-arousal) [48] to describe facial expressions, but very few classifiers have been developed to use it as of yet [22]. An even different approach is to represent facial expressions as a combination of various basic ones instead of belonging to a single class [49], but this representation has not yet become very used.

Table 1.1 reports some basic information about the currently best classifiers for the dataset that were considered in this project, as described in Section 2.1.1.

Table 1.1: *Current state of the art*

Dataset	Classes	Classifier type	Depth	Accuracy	
CK+	6	CNN	?	98.9%	[46]
	7	CNN/DBN+SVM	5	93.70%	[50]
	8	CNN/DBN+SVM	5	92.05%	[47]
AffectNet	8	CNN (AlexNet)	8	68%	[22]

Chapter 2

Methodology

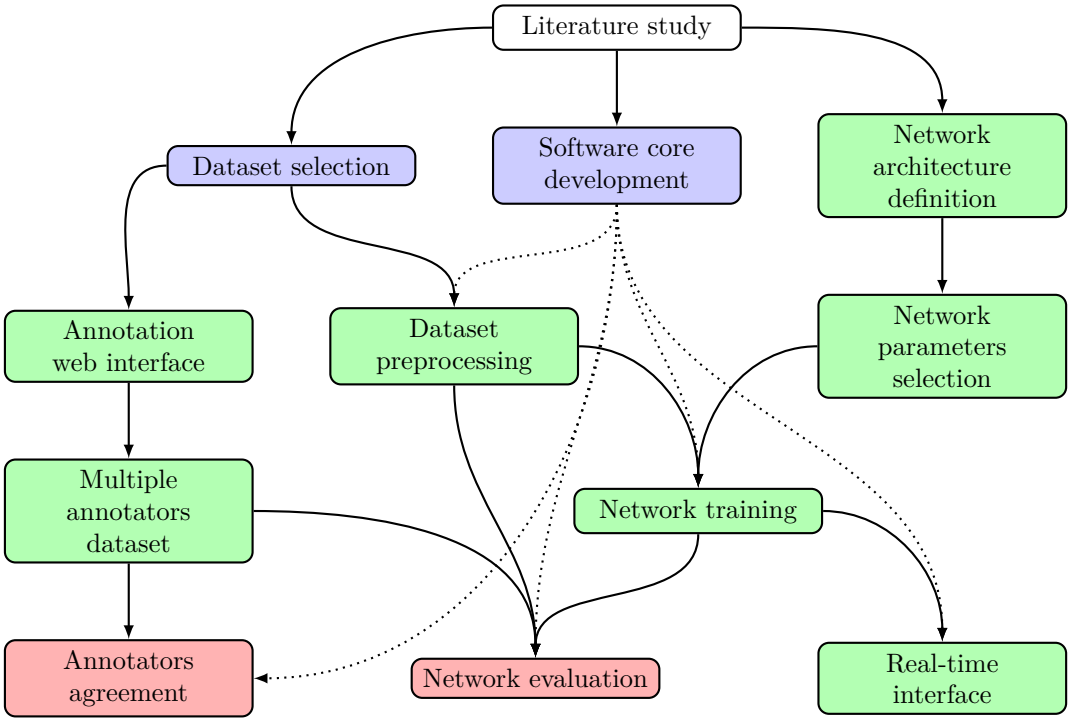


Figure 2.1: *Diagram of the various phases of this work. The arrows show which phases needed to be completed before working on another. The dotted arrows are only used for clarity. The colors indicate in which section each phase of development is treated:*

- blue*: Section 2.1*
- green*: Section 2.2*
- red*: Section 2.3 and Chapter 3*

As explained in Section 1.2.1, the goal of the project was to exploit deep learning techniques (in particular using artificial neural networks) to develop a classifier for FER that could be run in real time, investigate whether or not this is a good approach for

facial expression recognition and which information (if any) could be used to improve the technique for this specific task. In order to accomplish this the work was divided in various phases, some of which could be worked on in parallel (this is summarized by Figure 2.1):

- literature study to find out what was needed for this task, what had already been done and what had not been tried before
- selection of the most suitable datasets (Section 2.1.1)
- development of the processing pipeline to get the images in the datasets to a normalized form suitable for training a neural network (Section 2.2.1)
- definition of a suitable network architecture for the classifier (Section 2.2.2)
- selection of the specific parameters to apply to that architecture in order to get various different classifiers to be analyzed (Section 2.2.2 and Appendix C)
- training of the neural networks on the previously selected datasets (Section 2.2.3)
- development of a real-time interface to test the capabilities of the classifiers in real life situations (Section 2.2.4)
- development of a web interface to re-label some of the available images using multiple annotators in order to build a new dataset that contained more information than what could be given by a single labeller, studying how this additional information could be used (Section 2.2.5)
- analysis of the acquired data, in particular regarding:
  - classification quality of the trained networks and interesting emerging patterns (Section 2.3.1)
  - computational cost of training the various networks (Section 2.3.3)
  - uncertainty in the definition of the emotion expressed by an image according to the gathered annotations (Section 2.3.4)
  - classification quality evaluated using the new information acquired with the help of the annotators (Section 2.3.5)

Most of this was supported by the development of a software framework that implemented all the important procedures and utilities needed to make this project possible. This framework is briefly described in Section 2.1.3.

2.1 Materials

2.1.1 Datasets

In the context of emotion recognition, various datasets have been released during recent years to allow for the task to be studied and automatic classifiers to be trained. Table 2.1 lists a selection of such datasets, comparing their main features. The datasets that were considered and further analyzed for suitability are:

JAFFE Japanese Female Facial Expression Database [51]

CK Cohn-Kanade AU-Coded Expression Database [52]

MMI MMI Facial Expression Database [53]

CK+ Extended Cohn-Kanade Dataset [54]

MUG Multimedia Understanding Group Facial Expression Database [55]

RaFD Radboud Faces Database [56]

FERG-DB Facial Expression Research Group Database [57]

RAVDESS Ryerson Audio-Visual Database of Emotional Speech and Song [58]

AffectNet [22]

In addition to still images, some database feature videos (RAVDESS and MMI) or image sequences (CK, CK+, MUG). FERG-DB, differently from the other datasets, uses CGI instead of images of real humans in an attempt to create prototypical facial expressions. There are also differences in the kind of annotations gathered. In fact, while most datasets contain categorical labels (most often 6 basic emotions and *neutral*), some (CK, MMI and CK+) add annotations of Action Units in the Facial Action Coding System (FACS) and AffectNet includes a dimensional model (continuous values of *valence* and *arousal*). Finally, they differ in the way the various emotions were elicited:

posed: the subjects were explicitly requested to express a certain emotion (JAFFE, CK, MMI, CK+, MUG, RaFD, RAVDESS)

induced: the subjects were not instructed to express a specific emotion, but it was induced through conversation or by making them watch an emotion-evoking video (CK+, MUG)

Table 2.1: *Summary of some Facial Expression databases*

Dataset	Data type	Annotation type	Condition	Selected
JAFFE	Still images	Categorical	Posed	×
CK	Image sequences	Categorical and FACS	Posed	×
MMI	Videos and still images	Categorical and FACS	Posed	×
CK+	**Image sequences**	**Categorical and FACS**	**Posed and induced**	✓
MUG	Image sequences	Categorical	Posed and induced	×
RaFD	Still images	Categorical	Posed	×
FERG-DB	Still images of stylized characters	Categorical	Generated	×
RAVDESS	Audio and video of speech and song	Categorical	Posed	×
AffectNet	**Still images**	**Categorical and dimensional**	**Wild**	✓

generated: it is the case for the CGI images of FERG-DB

wild: the images were not created for the specific purpose of assembling a facial expression database but were gathered from different sources so to build a dataset of images as would be found naturally in various circumstances (AffectNet)

For this project, the choice of datasets was eventually restricted to AffectNet and the Extended Cohn-Kanade Database.

AffectNet

This dataset was assembled by querying three major search engines for images corresponding to 1250 emotion related tags and automatically detecting suitable faces among those images. Each image was then annotated with an emotion label (neutral, happiness, sadness, surprise, fear, disgust, anger, contempt, none, uncertain, no-face) and a valence/arousal pair. Half of the images were annotated by 12 human annotators and the other half by an automatic algorithm. The dataset is also already divided in a training and validation set.

Table 2.2: *Absolute frequencies of the various classes in the original AffectNet dataset and in the used dataset*

	AffectNet		**Used dataset**	
Emotion	**Training**	**Validation**	**Training**	**Validation**
Anger	24882	500	22904	468
Contempt	3750	500	3623	479
Disgust	3803	500	3632	474
Fear	6378	500	5850	461
Happiness	134416	500	129609	479
Neutral	74874	500	70751	474
Sadness	25459	500	23381	472
Surprise	14090	500	13272	467
None	33088	500	0	0
Uncertain	11645	500	0	0
No-face	82415	500	0	0
Total	414800	5500	273022	3774

The AffectNet dataset was selected to be used to train the neural networks because of its "wild" nature. Since the contained images were gathered from various sources, they had differing sizes, light conditions, sex, age and ethnic group of the subjects, image qualities, face orientations and "level of spontaneousness" (some images were clearly posed, while other show candid shots of genuine emotions). These characteristics make this dataset the perfect fit for a neural network that should work with webcam images taken in wildly different conditions. The great number of examples is also suitable for Deep Learning purposes.

The network was trained and validated only on those images that:

- were manually annotated by a human
- contained one of the seven basic emotions included in the dataset or the *neutral* label
- contained exactly one face according to the normalization algorithm (see Section 2.2.1)

The number of examples for each emotion in the final dataset can be seen in Table 2.2 and their relative frequencies in Figure 2.2.

By looking at the relative frequencies for the training sets in Figure 2.2, it is clear that some classes are over-represented compared to others, so training the network without any counter-measure would lead to overfitting caused by a preference for the *happiness* and *neutral* labels. To prevent this, the batches of examples used to train

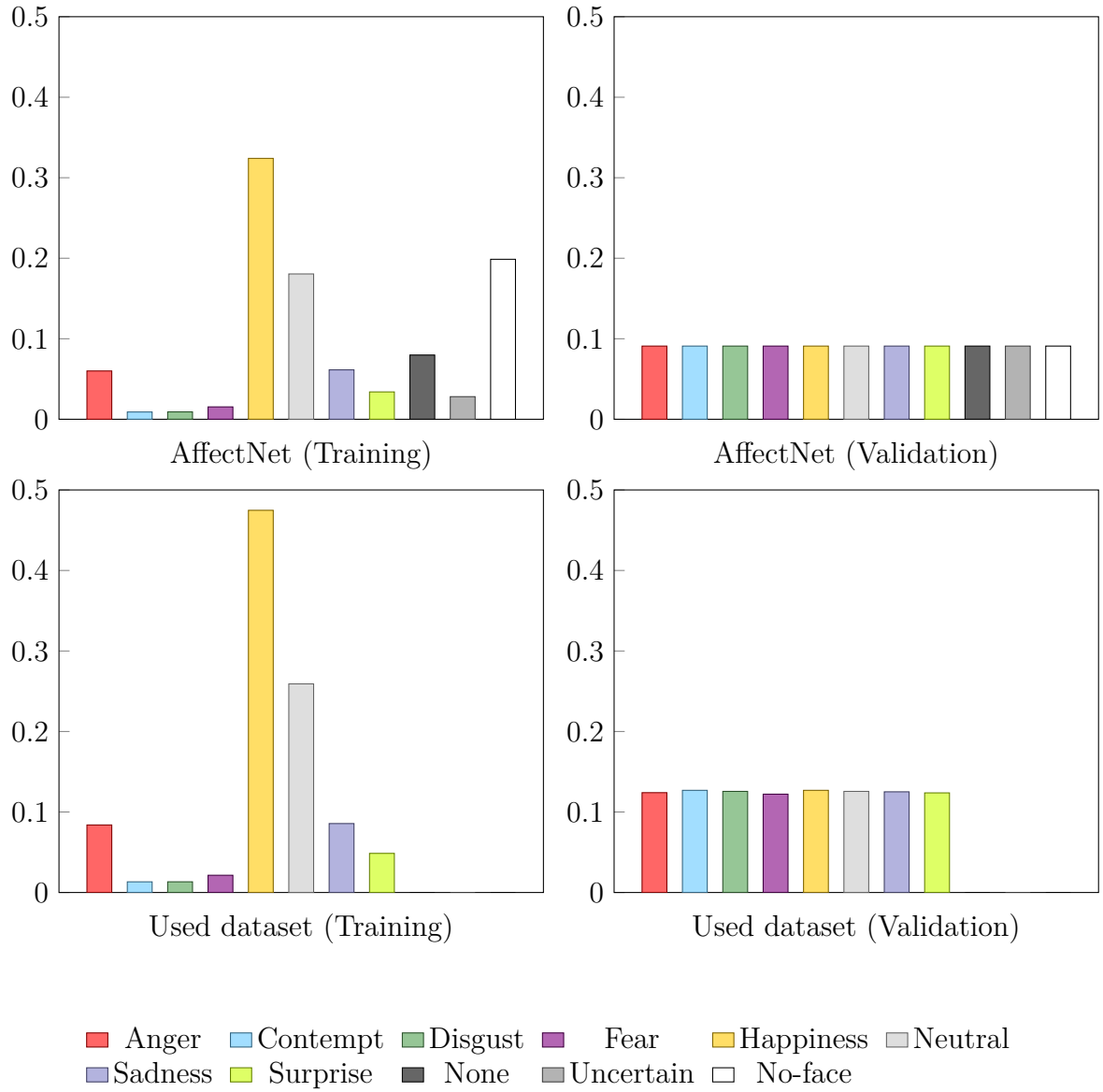


Figure 2.2: *Relative frequencies of the various classes in the original AffectNet dataset and in the used dataset. The imbalance in the training sets can be clearly seen.*

the network were assembled by taking an (almost, depending on the batch size) equal number of examples from each class.

Extended Cohn-Kanade

The Extended Cohn-Kanade Dataset, together with the previous Cohn-Kanade database, is one of the most used datasets in the context of emotion recognition. Just like its predecessor, it contains various sequences of images where subjects start from a neutral expression and then show a requested expression (among *Anger*, *Contempt*, *Disgust*,

Table 2.3: *Absolute frequencies of the various classes in the CK+ dataset*

Emotion	Images	Images (validation)
Anger	45	18
Contempt	18	18
Disgust	59	18
Fear	25	18
Happiness	69	18
Neutral	593	18
Sadness	28	18
Surprise	83	18
None	0	0
Uncertain	0	0
No-face	0	0
Total	920	144

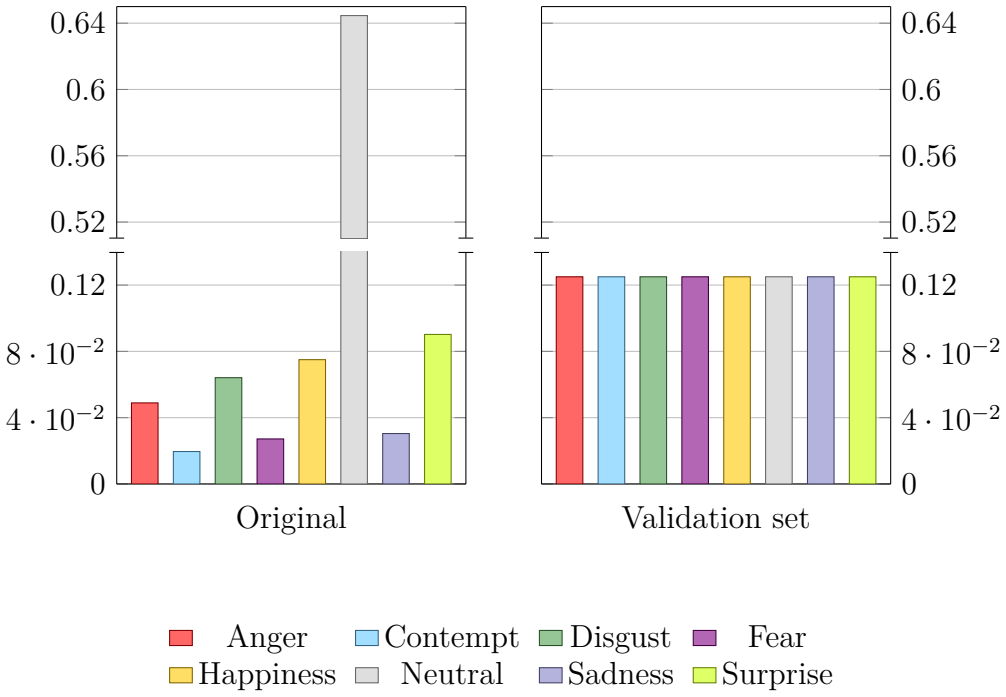


Figure 2.3: *Relative frequencies of the various classes in the CK+ dataset. The* neutral *label is much more represented in the original dataset because all sequences (included the non-labeled one) started with a neutral expression.*

Fear, *Happy*, *Sadness* and *Surprise*, the same emotions used by the AffectNet dataset) reaching the apex of expressivity in the last frame. All frames are FACS coded and the performed expression are labeled (differently from the original CK dataset that indicated the *requested* expression).

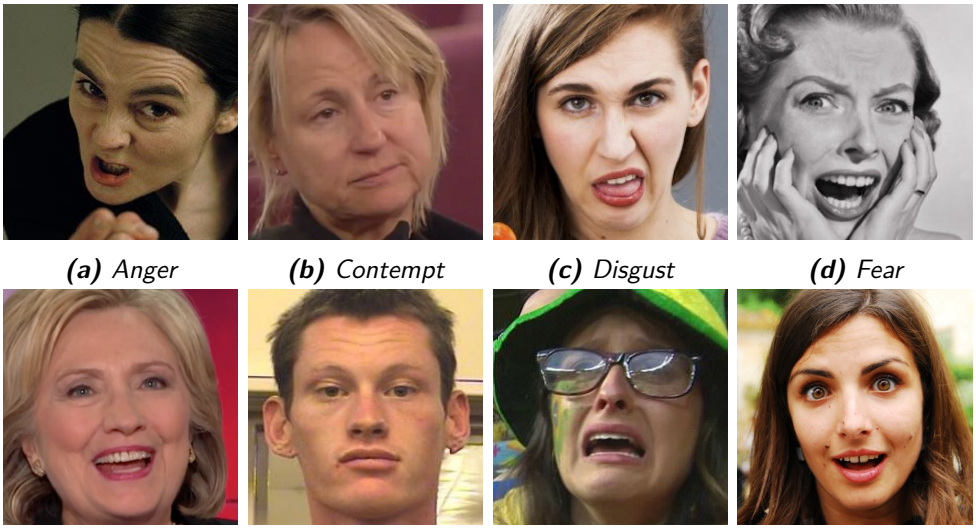
The sequences show the subjects from the front with a good illumination and the shown expressions closely follow the stereotypical rules described in the FACS Investi-

gators Guide [59]. Because of this and the fact that it is the most standard dataset in this field, it was chosen to use the CK+ dataset for further validation of the trained classifiers and to check how they perform on "easier" tasks.

For each sequence, only the first frame (labeled as *neutral*) and the last one (if an emotion label was available in the dataset) were taken into account, ignoring the transition frames. Since not all emotions appeared with the same frequency, a validation set was extracted by randomly picking images from each class so to have the same number for all classes.

Table 2.3 and Figure 2.3 show the absolute and relative frequencies of the various labels in the dataset.

Facial expression classes



(a) *Anger* ***(b)*** *Contempt* ***(c)*** *Disgust* ***(d)*** *Fear*

(e) *Happiness* ***(f)*** *Neutral* ***(g)*** *Sadness* ***(h)*** *Surprise*

Figure 2.4: *Facial expression classes considered in this work*

As stated previously, the used classes are: *anger*, *contempt*, *disgust*, *fear*, *happiness*, *neutral*, *sadness* and *surprise*. Examples are shown in Figure 2.4. These categories were chosen because:

- they are generally considered to be culture-independent universals of facial expression [60][61][62]
- they are the only labels in the AffectNet dataset that actually represent specific expressions
- they are the labels also used by the CK+ dataset

2.1.2 Hardware

The whole project (in particular the network training) was done using a Dell laptop with the following technical specifications:

Address space: 64 bits

Main processor: Intel Core i7-6500U, 4 cores running at 3.1 GHz

RAM: 2 DDR3 banks storing 4GiB each

Graphical processor: AMD Radeon R5 M335, a low end GPU

The computer also had a webcam that was used for testing the real-time applications.

2.1.3 Software framework

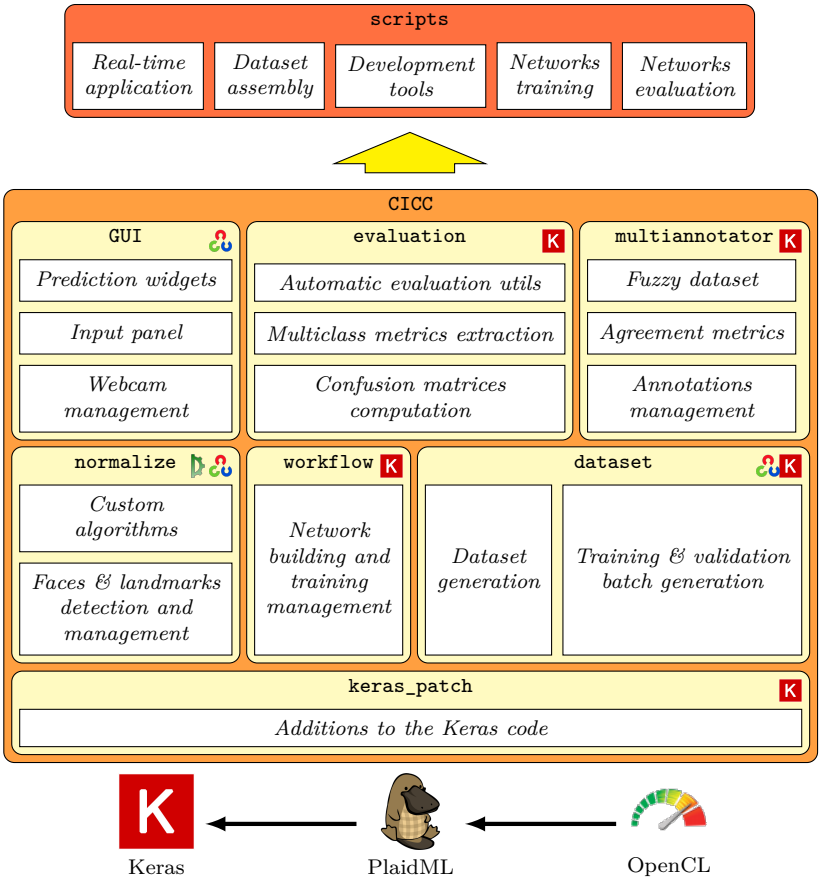


Figure 2.5: *Architecture of the developed software. Each box indicates one of the main software components of this project. Higher level components depend directly on the components below them and on the external libraries shown by the following icons:* K *Keras* *Dlib* *OpenCV*

Almost all of the phases in Figure 2.1 needed an extensive use of automated algorithms and software interfaces to be completed. For this reason, one of the first steps (which continued until the end of the project) was to develop a suitable software framework on top of which all the needed procedures and applications could be built.

All the software was written in Python 3.7.0[1], except for the website described in Section 2.2.5 which was written in PHP 7[2] with a MySQL 5.6[3] database. Python was chosen for its flexibility, ease of use and speed of development. Various C++[4] libraries (mostly in the form of Python wrappers) were also used for the most computationally intensive tasks, in addition to other Python packages that use highly optimized code. A short description of the most important ones is reported at the end of this section.

The software core of this project formed a Python package that was called *Complete Interface for Complex Classifications* (CICC for short). Figure 2.5 shows the structure of this package with its main components and dependencies on external libraries. Almost all of the code in the CICC package also depends on modules from the Python Standard Library and the SciPy stack.

Since neural networks were chosen as the main technology for the classifiers, a suitable library had to be used. The Keras Python package was selected for its simple high-level interface and its capability of accelerating the elaborations using available GPUs for the main processing.

The software for this project was developed on the PC described in Section 2.1.2 which mounted both a Linux distribution (latest Arch Linux release at the moment of publication) and Windows 10. The code was written and tested using partially both operating systems, except for the actual neural network training that was performed solely using Windows 10. The only support for AMD GPUs, in fact, is through the use of the OpenCL standard, but the support of this hardware on Linux is currently quite limited. The PlaidML backend for Keras was thus selected since it is the only one supporting OpenCL.

Following, the various used packages (starting with the one developed in this project) are described.

Core package

CICC (Complete Interface for Complex Classifications) is the Python package that was specifically developed in this project as a base for all subsequent phases. It implements most of the technical and mathematical solutions presented in this work and

[1] `http://www.python.org`
[2] `http://www.php.net`
[3] `http://www.mysql.com`
[4] `http://isocpp.org`

defines various utilities that can be useful not only for this project but for many similar applications. The specific implementation and full documentation of the package is out of the scope of this work and the source code of the package can be found at `https://gitlab.com/udscbt/cicc`.

The main components of this package are:

- `keras_patch`: adds features to the Keras package, in particular it enables the use of multidimensional metrics during network training and evaluation and defines custom metrics, activation and loss functions needed for this project
- `normalize`: provides utilities for the detection of faces and landmarks (based on the `dlib` library) and implements the basic algorithm for the normalization of the images of the dataset
- `dataset`: manages the creation of datasets suitable for this project from badly organized images and labels data and includes classes to be used with Keras to extract well-formed batches from those datasets
- `workflow`: provides a way to organize the neural networks indicating the architecture, layer parameters and training hyperparameters and to define a workflow to follow to automatically train and evaluate the various networks
- `evaluation`: implements the evaluation metrics used to quantify the classification quality of the networks together with utilities to easily apply them to the networks inserted in the workflow
- `multiannotator`: provides support for annotations data from multiple annotators, implements various metrics related to this and extends the `dataset` package to datasets with fuzzy/probabilistic labels
- `GUI`: offers various utilities for the development of graphical components and applications that interface with multiclass image classifiers, in particular providing simple classes to manage a connected webcam, show images taken from the filesystem or that same webcam and augment them with information such as landmarks position and classification predictions

This package is released under the GNU General Public License version 3.

Keras

Keras[5] is a Python package for building, training and evaluating neural networks. It must run on top of a suitable computational library (TensorFlow[6], CNTK[7] and Theano[8] are officially supported) and it speeds up the development by providing a high-level API that allows to focus more on the network architecture and parameters rather than on implementation.

The main features of Keras are:

- code independent from computational backend
- consistent and simple APIs
- clear and actionable feedback upon user error
- modularity
- easy extensibility
- no need for external files other than Python source code
- implementation of most standard network architectures (both convolutional and recurrent)
- support for both CPU and GPU

It is released under the MIT license.

PlaidML

PlaidML[9] is an advanced cross-platform tensor compiler written in C++ and Python that enables the development and execution of deep learning applications especially on devices with computing hardware not well supported by most of the other similar libraries. Even if it is not officially supported, it can be used as a backend for the Keras APIs accelerating training workloads with customized or automatically-generated Tile code[10] In addition to Keras, it supports ONNX[11] and nGraph[12].

[5] `http://keras.io`
[6] `http://www.tensorflow.org`
[7] `http://www.microsoft.com/en-us/cognitive-toolkit`
[8] `http://deeplearning.net/software/theano`
[9] `http://github.com/plaidml/plaidml`
[10] *Tile* is a language used by PlaidML to define the operations to be performed by the GPU.
[11] `http://onnx.ai`
[12] `http://ngraph.nervanasys.com/index.html`

While OpenCL support for TensorFlow and Theano was announced but never actually completed, PlaidML is the Keras backend that supports the largest number of GPUs by using OpenCL and thus not requiring the Nvidia proprietary frameworks CUDA[13] and cuDNN[14], while achieving comparable performance.

It is released under the Apache license version 2.0.

OpenCL

OpenCL[15] (Open Computing Language) is an open standard for cross-platform, parallel programming of diverse processors found in a variety of devices. It defines a kernel language based on C++14 that enables the use of the parallel computing capabilities of CPUs, GPUs, DSPs, FPGAs and other hardware accelerators, thus making it possible to use the same code on wildly different platforms.

It is released under a license similar to the MIT license.

SciPy

The SciPy[16] stack is a collection of Python packages and related software that has over the years become the de facto standard library for scientific computing with Python.

Its core packages are:

NumPy manages N-dimensional array objects, providing powerful linear algebra functionalities

SciPy provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization

Matplotlib produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms

All the packages are licensed separately, but mostly with the BSD license.

OpenCV

OpenCV[17] (Open Source Computer Vision Library) is a cross-platform library for computer vision and machine learning that includes more than 2500 optimized algorithms performing a variety of advanced tasks, ranging from face detection to 3D point cloud

[13] `http://developer.nvidia.com/cuda-toolkit`
[14] `http://developer.nvidia.com/cudnn`
[15] `http://www.khronos.org/opencl`
[16] `http://www.scipy.org`
[17] `http://opencv.org`

generation from stereo cameras. It's written in optimized C++ that supports parallelization of the workload to increase efficiency, but Python, Java and MATLAB interfaces are also available.

It is released under the BSD license.

Dlib

Dlib[18] is a C++ toolkit containing advanced machine learning, numerical analysis and image processing algorithms together with utilities for more technical aspects of software development. It also includes state of the art pre-trained models for object detection and other advanced tasks.

A Python wrapper package is available to access the C++ functions.

It is released under the Boost Software License.

2.2 Development

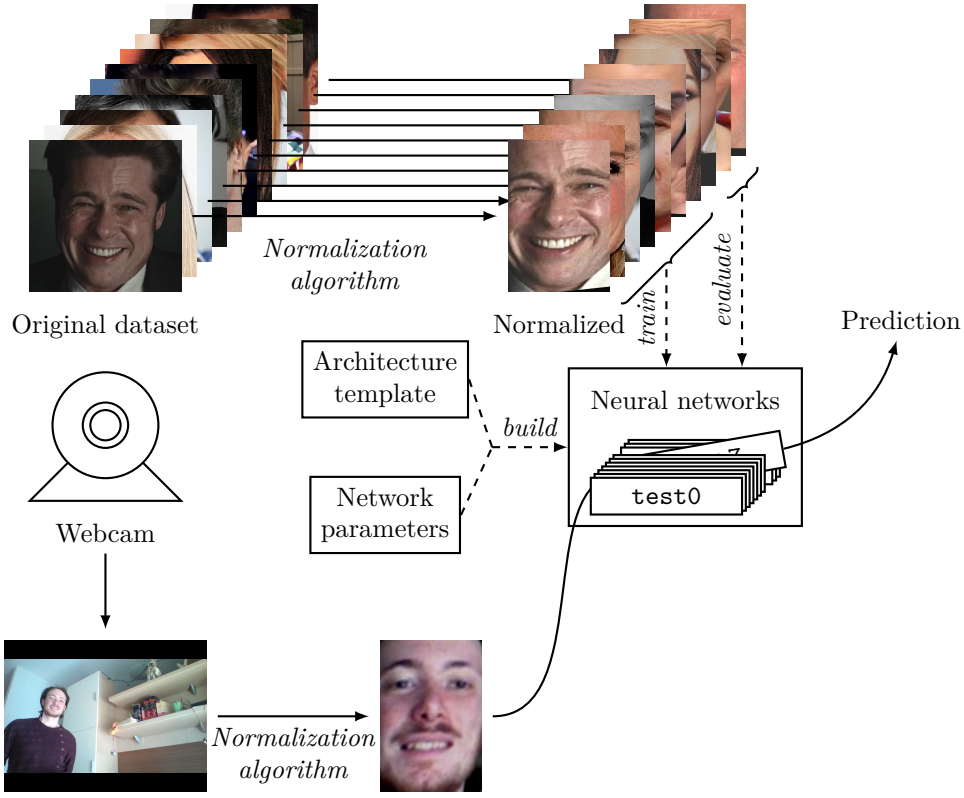


Figure 2.6: *Scheme of the developed system*

Figure 2.6 shows a schematic representation of the various parts that form the

[18] `http://dlib.net`

completed project and how they fit together. In the end, a system was developed to train and evaluate neural networks on images showing facial expressions and use the trained networks to classify webcam images in real time.

The AffectNet dataset contains faces at various scales, orientations and positions inside the images (just like it would happen with images taken from a webcam). The neural network would hence need to waste a lot of computing resources to account for all the possible face localization differences in the input images. To avoid this, the input images (both in the dataset and during actual use of the network) are preprocessed by a normalization algorithm that aligns the detected faces to a standard template, at the same time removing useless information like hairs and background. This algorithm is described in Section 2.2.1 and was applied to the CK+ dataset too.

The network topology then needed to be defined (Section 2.2.2) and the various networks, with differing parameters, trained (Section 2.2.3). In the choice of the parameters, the dataset used for training also had to be taken into account since the CK+ dataset is much simpler and smaller (thus more prone to overfitting) than the AffectNet dataset.

While the numerical results reported in Chapter 3 are useful in quantitatively evaluating the networks, it is also important that those same networks could actually be used for real applications. For this reason, an application was created to test the capabilities of the networks in real-life situations. The goal was to create a simple interface to acquire images from a webcam, process them the same way as the images taken from the dataset and then show the resulting prediction of the trained classifier. This application is described in Section 2.2.4.

Separately from all of this, the possibility of using multiple annotators to have more data about the emotion expression of each image was considered. Section 2.2.5 describes how a dataset of this kind was assembled and a way to use it to train neural networks was studied.

2.2.1 Normalization algorithm

The proposed algorithm is similar to the one found in [63], but uses more advanced feature detection algorithms and a slightly different approach at face alignment. It processes the image in the following steps (from step 2 onward, the operations are performed on each detected face separately):

1. all faces in the image are automatically detected and cropped to their bounding box[19]

[19]The standard detector from the `dlib` library was used. It uses a fixed sized sliding window

2. 68 facial landmarks are located using another pre-trained detector[20]

3. 3 markers are computed as linear combinations of the positions of the landmarks

4. the image is morphed through an affine transformation that places the 3 markers in the corresponding positions of the template (at the same time, the image is also resized to the input shape of the network)

5. optionally, the image is converted to grayscale

6. the image colors are normalized

To speed up the processing of large images, the algorithm can optionally use a subsampled version of the image for locating the face and the facial landmarks and then use the detected features to normalize the full resolution image. Even if no subsampling is performed, the image is always converted to grayscale during detection.

A graphical user interface was developed to manually determine the optimal parameters for this algorithm. In the end, it was chosen to use as the three markers the average positions of the landmarks corresponding, respectively, to the two eyes and the mouth. The color correction is made with the following formula (separately for each channel if the image wasn't converted to grayscale):

$$y = 128 \left(1 + \frac{x - \mu}{2.5\sigma}\right)$$

where x and y are respectively the original and new color value, μ is the mean color value of the original image and σ is its standard deviation. The obtained value is finally clipped between 0 and 255. Other color normalization methods were considered, in particular:

- **histogram equalization**: it had the side effect of creating artifacts that could be wrongly interpreted by the neural network
- **linear normalization**: it proved less effective than the proposed method when people of different skin color were considered

A screenshot of the interface and more information about the selected parameters can be found in Appendix A.

Figure 2.7 shows an example of the application of this algorithm.

linear classifier over a Histogram of Oriented Gradients image pyramid as discussed in [64] with the modifications described in [65].

[20]The standard detector from the `dlib` library was used. The detector implements the algorithm described in [66] and it was trained on the iBUG 300-W face landmark dataset [67].

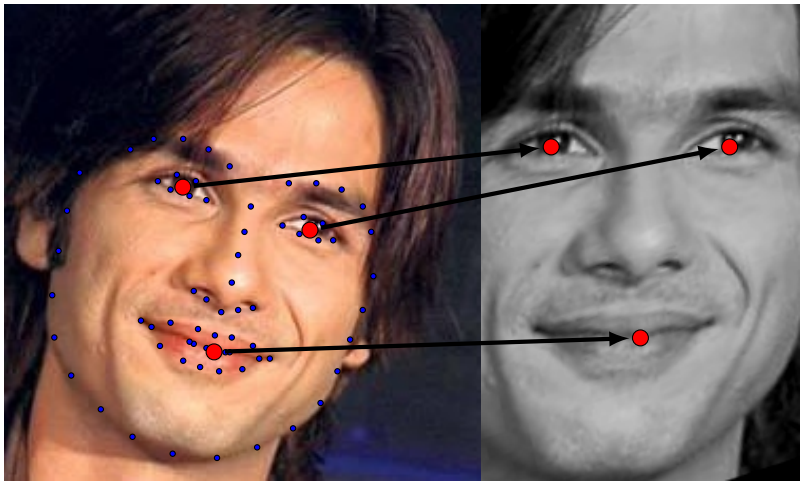


Figure 2.7: *Normalization process. On the left the original image with superimposed landmarks and fiducial points, on the right the normalized image with the corresponding fiducial points.*

The effectiveness of the algorithm in aligning the key facial features can be seen in Figure 2.8 and Figure 2.9 relatively to the dataset images and a real life application respectively. In both of these figures, it is quite evident that the features position in the not-normalized images is very inconsistent between the samples (less so for the AffectNet images since they are already obtained from a crop around detected faces), while the normalized samples are consistent enough to allow for the averaged image to show distinct features relative to emotion (Figure 2.8) or subject (Figure 2.9). In the latter case, in particular, the alignment is so consistent that the resulting image almost looks like a single still frame.

2.2.2 Network architecture

For the network architecture, various structures have been analyzed, all following a basic template common among neural networks working on image recognition. Figure 2.10 shows this template:

- input layer
- various *Convolution-Maxpooling-Rectified Linear Unit* (CMR) blocks
  - parallel convolution layers with different kernel sizes (extracting features at different scales but at the same abstraction level)
  - max pooling layer to reduce the size of each map while keeping track of high activation values
  - rectification to ignore negative activations

Figure 2.8: *Each image is the average of 100 samples taken from a single class. For each pair, the image on the left was made with samples taken from the AffectNet dataset (the images were resized to a fixed shape before averaging) while the image on the right used the same samples after normalization. The classes are (clockwise from the top): contempt, anger, surprise, sadness, neutral, happiness, fear, disgust. The central image is the average of all the classes.*

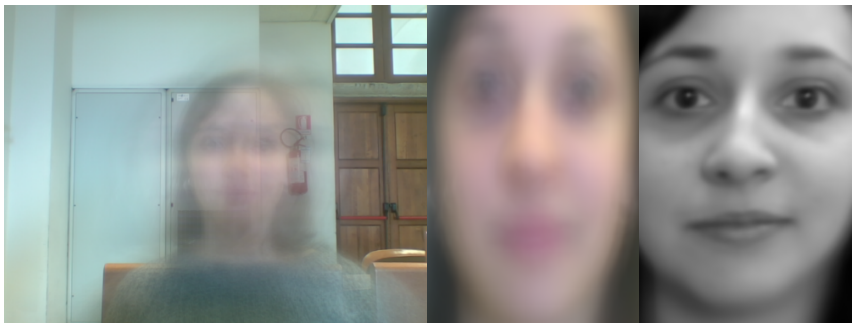


Figure 2.9: *Each image is the average of 100 samples taken from a video captured by a webcam. On the left the raw images were used, in the center face detection, cropping and resizing were performed, and on the right the images were normalized before averaging.*

- flattening of the feature maps to a linear array for the next layer
- fully connected layer combining the information contained in the various feature maps
- final softmax layer performing the classification

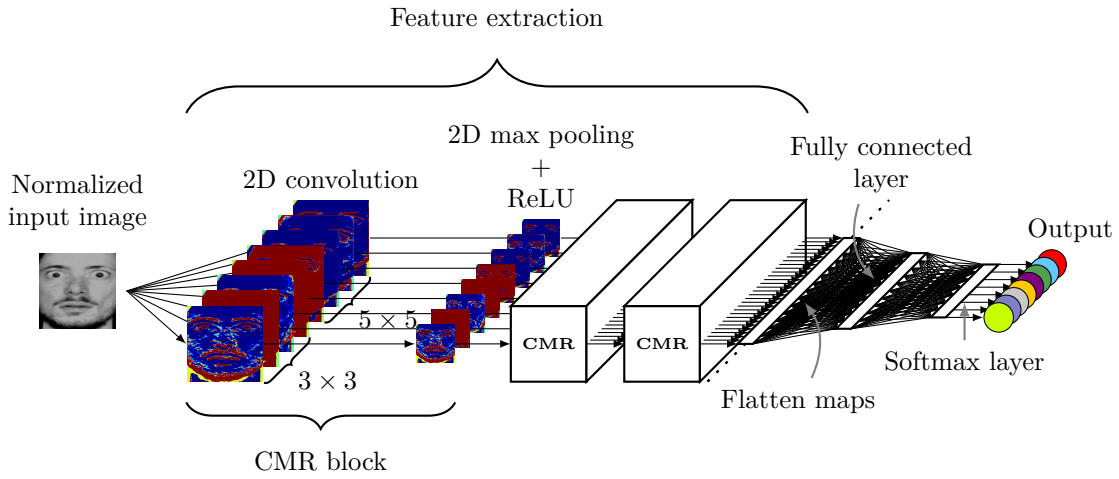


Figure 2.10: *Template architecture of the networks*

Starting from this template, the network depth and the parameters of the various layers were changed in an attempt to find the optimal network architecture for this task, also with the help of a graphical interface described in Appendix B. While the developed infrastructure used to create and train the various networks allowed for the definition of all parameters of the various layers, the following choices were made for all trained networks:

- convolution layers had kernels of size either 3 or 5
- convolution layers had stride length equal to 1 in both directions
- convolution layers had a sigmoid activation function

The kernel size choice was determined by the will to keep the number of parameters of each filter quite low, so that they would only work with simple templates and higher level features would be detected by deeper layers instead of having highly specialized filters that could incur in overfitting. The stride length was fixed to one so to avoid loss of information and have a bit of superposition of the masks used by the convolution layer. Finally, the activation function was chosen according to the accepted standard to have bounded non-linear activation functions in the feature-extracting layers:

- bounded because the output should represent a level of activation, so arbitrarily high numbers are not appropriate
- non-linear because the filters should perform operations on the input that can not be accomplished by a simple linear combination

The most common activation functions of this type (and the ones natively supported by Keras) are: hyperbolic tangent, sigmoid and hard sigmoid. Among these the sigmoid

function was selected, being the most commonly used one, knowing that all three are very similar.

Generally, each CMR block had more 5×5 filters than 3×3 ones because of the higher number of parameters of the former, which meant that more useful filters could be found. The 3×3 filters are in fact a subset of the 5×5 ones, but they were included separately in the network structure in order to force the network to find simpler features too. The color information was mostly ignored since the first tests showed no significant difference in accuracy when the image was converted to grayscale first while the reduction of channels allowed for bigger networks to be built without affecting performance.

Most networks were trained on the AffectNet dataset, but a few were also trained using data from the CK+ dataset in order to make a comparison with the results that could be go gotten with a simpler dataset. These latter networks were kept very small in order to prevent overfitting, since the CK+ dataset is quite small and simple. To further avoid overfitting, two dropout layers (one before and one after the fully connected layer) with a rate of 0.5 were added to these networks.

In Tables C.1 to C.4, details about the parameters used for each trained network are shown. The networks which name starts with "`test`" were trained using the AffectNet training set described in Section 2.1.1, while the ones which name starts with "`Compare`" were trained on the CK+ dataset described in Section 2.1.1. Note that `test15` appears to be without any layer according to the tables. This is because this network wasn't built from scratch but it is simply a copy of `test14` that was trained again in a different way (see Section 2.2.3 for more information about how the networks were trained). Almost all convolutional layers had a dilation rate of 1, except for half of the filters in the second CMR block of `test24` and `test25`. The `Compare4` network also had a stride length of 3 in its CMR layer, differently from all other networks.

2.2.3 Network training

Most of the networks were trained in a supervised manner using the data from the dataset described in Section 2.1.1 which was extracted from the AffectNet dataset. Since the training set is very large (no data augmentation was deemed necessary) and the available computational resources were limited, it was chosen not to train the networks on the whole dataset (273022 images) at each epoch but instead feed the training algorithm 10000 samples per epoch, trying to use different images each time. This last requisite was however not always satisfied in order to avoid overfitting on the most represented classes (Section 2.1.1 clearly shows that the *happiness* and *neutral* classes are way over-represented in the dataset). In fact the images were not simply

taken in the order in which they were stored, but the algorithm in Figure 2.11 was used to generate batches of images that ensured equal distribution of all the labels and at the same time reduced the risk of overfitting on few images.

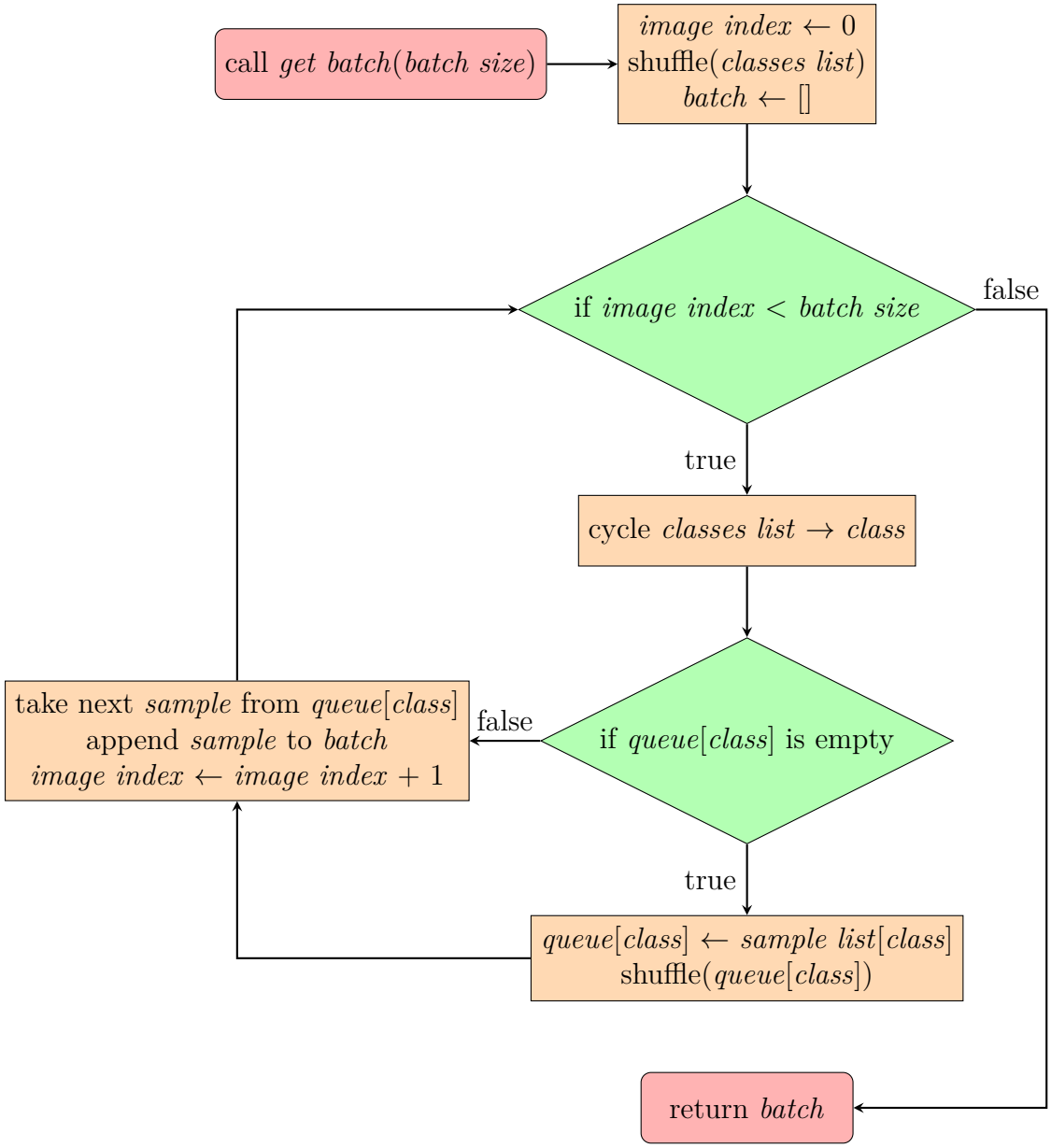


Figure 2.11: *Flowchart of the algorithm developed to generate the batch of images used to train the neural networks. Here "*sample list*[*class*]" and "*queue*[*class*]" are initialized at the start of the training to be the list of samples belonging to a specific category, while "*classes list*" is the list of those categories.*

Four neural networks were instead trained on the CK+ dataset described in Section 2.1.1. In order to balance the various classes, considering the small size of the dataset, it was chosen to oversample the less represented classes. At each epoch, $83 \cdot 8 = 664$ images were presented to the network using the algorithm in Figure 2.11.

This way, all the images of class *surprise* were shown exactly once per epoch, different images for the *neutral* class were used at each epoch (until they started repeating after $593 : 83 \approx 7$ epochs) and all other classes had repeated images to reach the right number of samples.

As with the network architecture, various training hyperparameters were tried in an attempt to find the best ones for this specific task. However some of those were fixed, in particular optimizer, loss function and evaluation metric.

The chosen optimizer was ADADELTA [68], an adaptive gradient-based optimizer derived from AdaGrad [69] but with substantial improvements in robustness, convergence speed at high epochs number and automatic hyperparameters update. It is a gradient descent optimization algorithm that changes a per-parameter learning rate based on the previous values assumed by the gradient for that parameter and the magnitude of the updates that were performed on that parameter, giving more importance to the most recent values.

The selected loss function was categorical cross entropy, which is the standard one used for multiclass classification tasks such as this one. It looks only at the network output corresponding to the correct label and penalizes misclassification with a loss growing very rapidly for small output values. It is defined for each example as:

$$CE(t, y) = -\sum_{c=1}^{C} t_c \log(y_c)$$

where y_c is the network output for class c, C is the number of classes and:

$$t_c = \begin{cases} 1 & \text{target label is } c \\ 0 & otherwise \end{cases}$$

Note that since the network output is computed by a *softmax* layer, y_c will never be equal to 0 and hence the logarithm is always defined.

Finally, the classification performed by the network was evaluated during training by measuring the accuracy, defined as:

$$A(T, Y) = \frac{\sum_{i=1}^{N} \begin{cases} 1 & max(T_i) = max(Y_i) \\ 0 & otherwise \end{cases}}{N}$$

where T is the list of target labels (as one-hot vectors) for all provided examples, Y is the list of network outputs for the same examples and N is the number of those examples.

A *minibatch* method was used, meaning that parameters updates were not performed only at the end of an epoch, but each time a smaller *batch* of data was provided to the training algorithm. Bigger batch sizes allow for smoother and less erratic updates, but are more prone to getting stuck in local minima. Bigger batches also speed up training, since fewer hard disk accesses are needed, reducing the overhead.

All network architectures (which parameters can be found in Appendix C) were trained 3 times for 100 epochs, varying the batch sizes to see how this affected training. A strategy that was tried was to split the training in two phases:

1. training with a big batch size, reducing the machine time necessary and quickly reaching a good starting point

2. training with a small batch size, trying to improve the previous optimization by exploring paths that couldn't be reached by the more consistent gradient descent of the previous phase

A similar approach was used for `test14` and `15`, where the first architecture was trained for 50 epochs with a big batch size and then a copy of the resulting model was used as the second network and trained for other 50 epochs with a different batch size. Even without this trick, the ADADELTA optimizer ensured that more finely tuned updates were performed when necessary.

Table C.6 shows the parameters used for the training. It must be noted that the batch size was partly limited by the computational power of the machine used, which had a limited memory and hence big batch sizes couldn't be used in conjunction with big networks.

2.2.4 Real-time application

The program was written in Python, using the Tkinter[21] GUI toolkit: it is the standard for Python applications and it is a thin object-oriented layer on top of Tcl/Tk[22]. No OS-specific functions were used, so it works indifferently on Linux, Windows, Mac OS or any other operating system supporting Python.

Figure 2.13 shows the interface of this application. As it can be seen from the *Image* menu, the program can work both with static images loaded from the filesystem and by capturing images in real time from a connected webcam. This image can then be navigated in order to focus on certain sections and ignore uninteresting parts (the network still "sees" the whole image). A trained neural network (saved as a Keras model

[21] `http://docs.python.org/2/library/tkinter.html`
[22] `http://http://www.tcl.tk/`

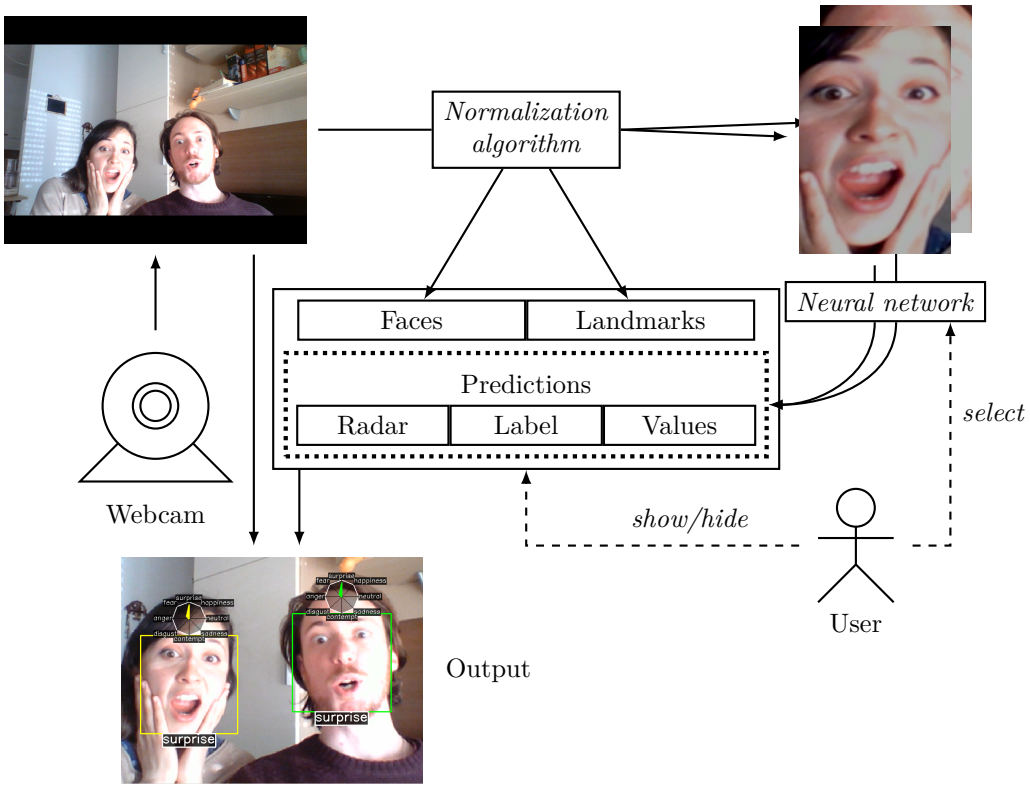


Figure 2.12: *Diagram of the real-time application working flow.*

in a `.h5` file) can be loaded into the program to test it and it will automatically try to guess the emotions shown by all faces in the image as soon as it appears, but it can also be explicitly instructed to make a prediction by clicking on the *Predict* button. The output of the network can be shown in three representations:

- radar chart
- single label
- list of values

The three representations can be separately enabled and disabled and their position relative to the faces can be selected by the user. In addition to the network output, the bounding boxes of detected faces and their facial landmarks can be shown.

Figure 2.14 contains screenshots from this application, showing various examples of use: indoor and outdoor, with good and bad illumination, with one or more faces, with and without glasses and with different features enabled.

Radar chart

If this feature is selected, it will display a radar chart showing the output value of the network for each emotion class, corresponding to the fixed chart axes. In addition to

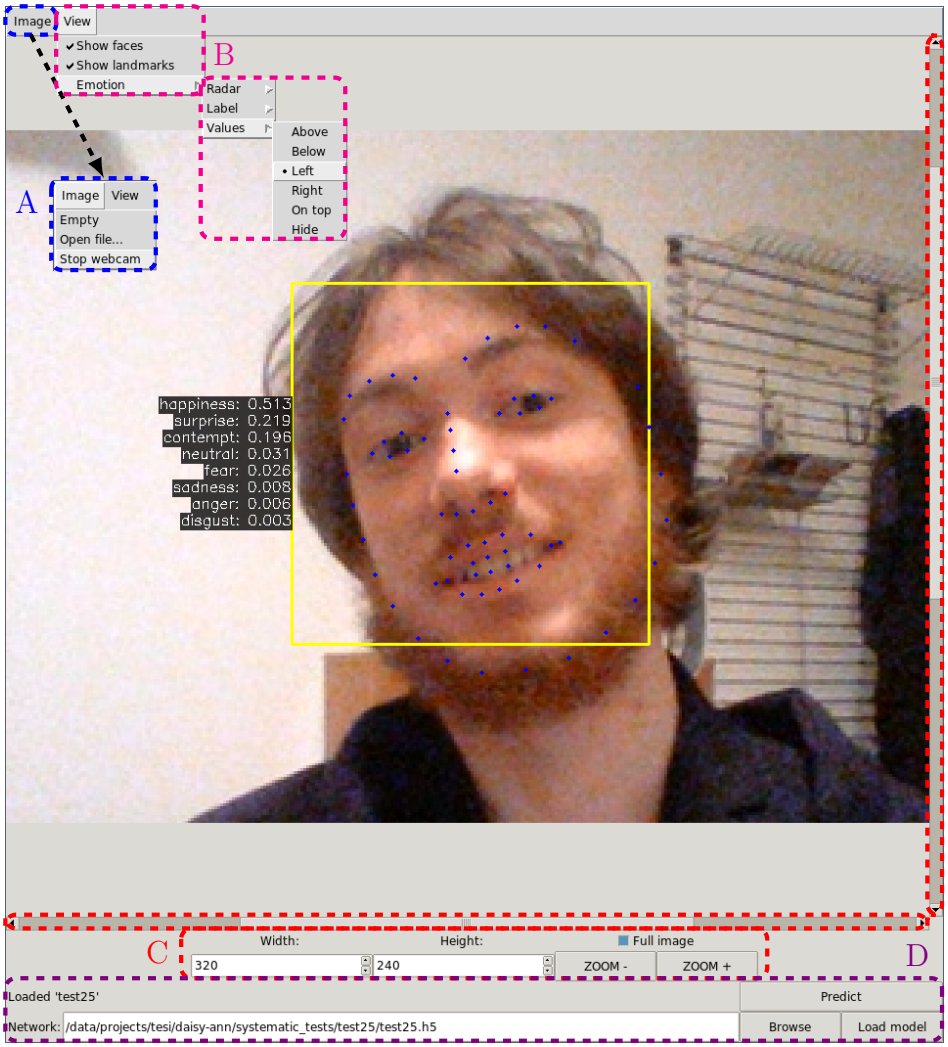


Figure 2.13: *Graphical User Interface of the real-time application.* ***A*** *is the menu used to select the input image;* ***B*** *is the menu used to turn on and off the various features;* ***C*** *are the controls to move around the image;* ***D*** *are the controls to load and use a neural network.*

the current output it also displays the previous outputs for that same face, shown more transparent the older they are, so that it is possible to see how the output is changing while the expression changes. In order to do this all faces are tracked independently, as shown by the different colors of the bounding boxes and radar charts.

Single label

This representation only shows the actual result of classification: which class has the higher output and thus to which class the image is predicted to belong. The selected class is simply displayed as a big text label showing the name of the class.

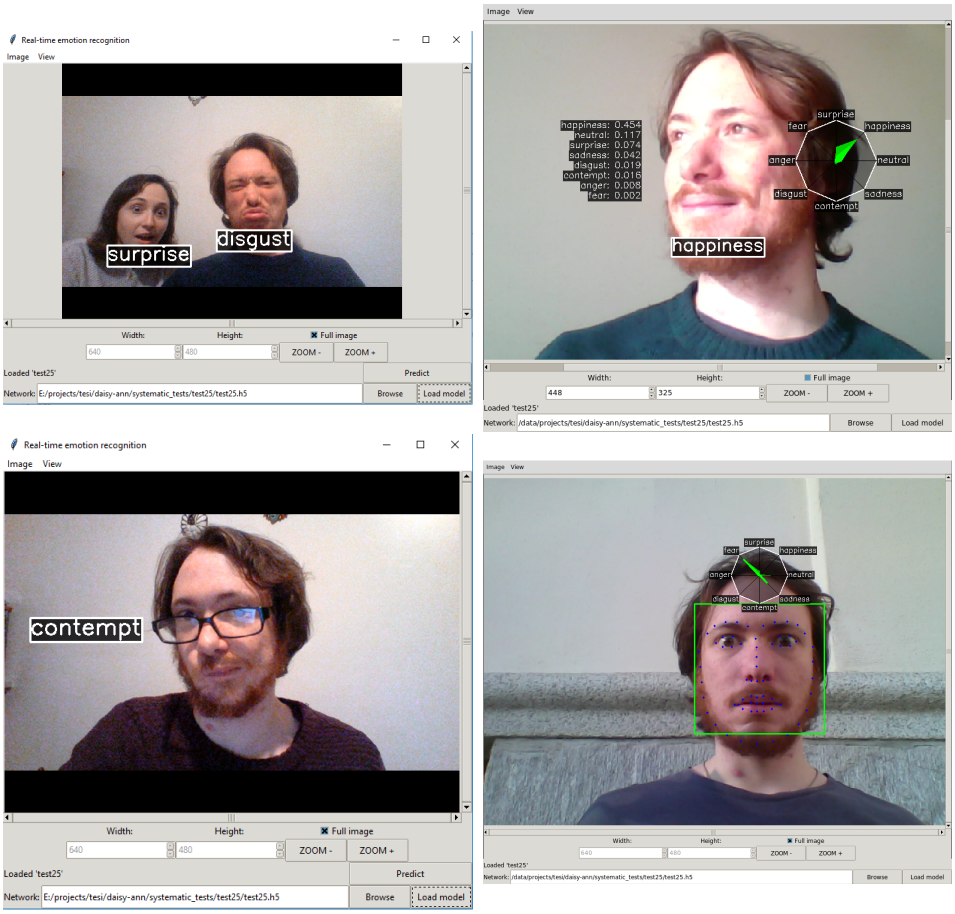


Figure 2.14: *Screenshots from the real-time application, using the* test25 *network. As can be seen, the interface is customizable, making it possible to turn on and off the various features, and works both under Linux and Windows. The pictures also show how the used network works smoothly in different environments, light conditions, image quality, face orientation and number of faces seen by the webcam, without problems related to the presence of glasses and beards.*

List of values

Finally, it is possible to display the numerical values of the output. The classes are ordered from the one with higher output to the one with the lower one and the output value is displayed next to the class name (with up to three decimals since more would only clutter the interface).

2.2.5 Multiple annotators dataset and network

While the AffectNet dataset is very useful because of its large number of examples and the fact that the images come from various sources making the dataset more general than similar ones, inspecting it shows that some of the annotations are quite questionable.

Figure 2.15 shows some selected images from the AffectNet dataset with their re-

spective expression labels. The main problem of these annotations is that the task of emotion classification is very subjective except for extreme cases, especially in this dataset where the images were taken *in the Wild* as described in [22], but each image was annotated by a single annotator. This problem was addressed by the original paper where it is reported that there was an annotation agreement of only 60.7% on 36000 images that were annotated by two annotators.

Figure 2.15: *Some examples of annotated images from the AffectNet dataset which show questionable labeling choices by the annotators.*

The choice of using only a single annotator for each image was made by the authors who preferred using few "expert" annotators instead of a crowd annotation service like Amazon Mechanical Turk which wasn't considered reliable enough. While this choice could be sensible for the annotation of valence and arousal[23], it is disputable that a crowd-sourcing approach could be more suitable for the categorical model. Some reasons are:

- emotion recognition is a inherent human ability, so the distinction between expert and naïve annotators is not that clear-cut
- the task at hand is quite subjective, especially for the more subtle expressions, so the opinion of a large number of people is more interesting than a single annotation
- some images can show ambiguous emotion expression for which it wouldn't be correct to say that only one label is "right"

[23]The original paper does in fact justify the choice with "annotating the valence and arousal requires a deep understanding of the concept"

- as shown both in Figure 2.15 and in the annotator agreement score reported by [22], using expert annotators does not make the annotation very reliable either
- [70] shows that multiple naïve annotators can perform as good as or even better than single expert annotators in emotion recognition in written text, which is a task quite similar to this one

For these reasons, the idea of using data from multiple annotators was explored. First, a small dataset was extracted from AffectNet and a website was created to gather annotations for it. This dataset was called WebAN from ***Web**-based relabeling of **A**ffect**N**et*. Then, various ways to use the acquired data to develop useful neural networks were studied.

Dataset (WebAN)

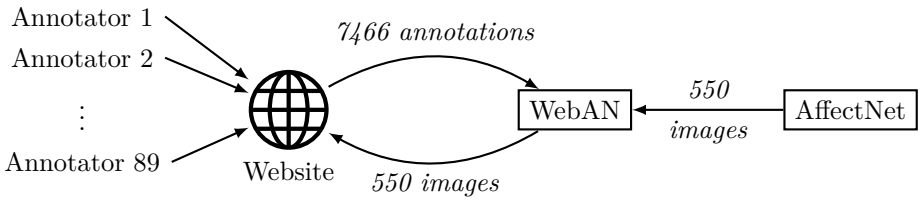


Figure 2.16: *Diagram of how the WebAN dataset was assembled*

From the original AffectNet dataset, 550 images were randomly extracted (50 for each class as annotated in the original dataset), but after the first selection two images had to be replaced. In fact, by random chance, three copies of the image in Figure 2.17 were selected[24] and only one was kept. Since two of them were labelled as *surprise* and one as *fear* (they were probably annotated by different annotators without anyone noticing the repetition), the *surprise* label was kept.

In order to gather the annotations, a website was created[25], making it accessible from anywhere and with many different devices, thus allowing for a more diverse annotator pool. The annotators were neither trained "experts" like the AffectNet annotators, nor random individuals trying to earn money doing simple tasks like with services like Amazon Mechanical Turk. Instead, various people were asked to annotate those images on a voluntary basis without any reward (so that only motivated annotators would participate, removing any ulterior motives and thus reducing the probability of annotators classifying the images randomly just to get through all images). Since no

[24]They were not completely identical, in fact one copy was at a slightly bigger scale and there were subtle differences due to the compression process, but nothing that could be detected by a human eye

[25]It can be found at `http://beethoven.altervista.org`

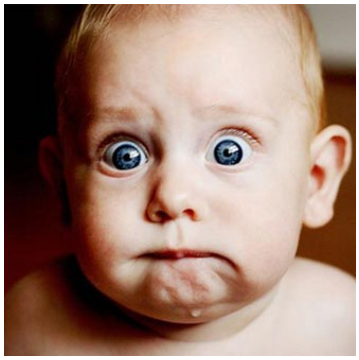


Figure 2.17: *Image that was found three times in the AffectNet dataset with differing labels:* surprise *twice and* fear *once.*

incentive was given for annotating the images except for the will to help in this research, very few annotators would have agreed to label too many images. Because of this, each annotator could annotate as many images as they wanted, while the website automatically presented the images that were annotated by the least number of annotators while never repeating the same image to the same annotator[26].

In the end, 89 annotators contributed to the project with 7466 annotations.

Figure 2.18 clearly shows that most annotators annotated only a little part of the dataset, while a few other annotated most or all of it. Figure 2.19 shows that presenting the least annotated images first allowed for the images to have a homogeneous number of annotations each.

The classes used matched the ones described in [22] so that the original dataset and the new annotations could be compared. Only the *uncertain* label was excluded, since this "uncertainty" could be better quantified by forcing annotator to use a specific label and measuring the agreement between annotators (low agreement indicates ambiguity).

The website interface was very simple and intuitive, so that it could be used by people without training other than a simple explanation when first starting to annotate. It simply showed the image to be annotated, the possible labels (with descriptions for less clear labels, already present in the initial explanation) and two buttons, one to submit the annotation and one to skip the image without annotating it. Figure 2.20 shows a screenshot from the website.

From the acquired dataset, all the images belonging to the *no-face* class had to be removed since they offer useless or even misleading information that is not suitable for training any kind of network. Even for this class, the agreement between annotators was not perfect, partially because not everyone understood what this category meant

[26]By repeating images, data could have been acquired and studied about intra-annotator variability, but this was deemed unnecessary for this project and it was preferred to have the annotators label more images rather than the same ones multiple times

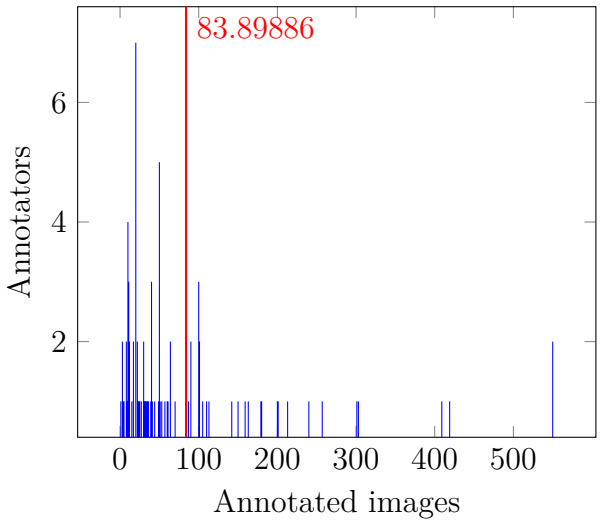


Figure 2.18: *Distribution of number of annotations per annotator. The bars height indicates how many annotators labeled a number of images equal to the one indicated in the horizontal axis. The red line shows the average number of annotations among all annotators. It can be seen that most annotators annotated less than a hundred images.*

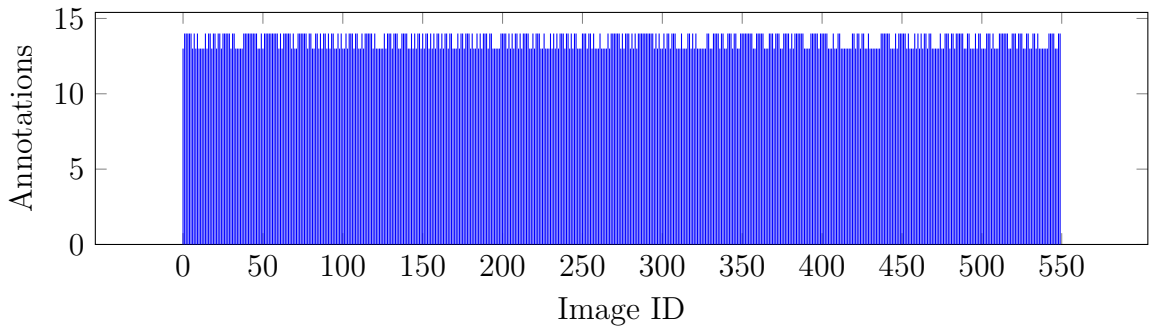


Figure 2.19: *Number of annotations per image. Each bar represents an image in the dataset and its height shows how many annotations were gathered for that image. The system ensured an equal distribution of annotations among all images as can be seen by the fact that all bars have (almost) the same height.*

and partially because people often tried to interpret emotions even when the image was not suitable for the task. To distinguish between usable and useless images, it was decided that images with 0 *no-face* annotations were considered reliable while images labeled with *no-face* more than 25% of the times were considered actually belonging to the *no-face* category. This criterion gives rise to the confusion matrix in Table 2.4 with respect to the original labels. The cases in-between were visually inspected for suitability, looking at the already normalized images so that only what was seen by the network was taken into account. At the end of this process, all remaining annotations for the class *no-face* were discarded.

Table 2.5 and Figure 2.21 show the classes distribution in the obtained dataset,

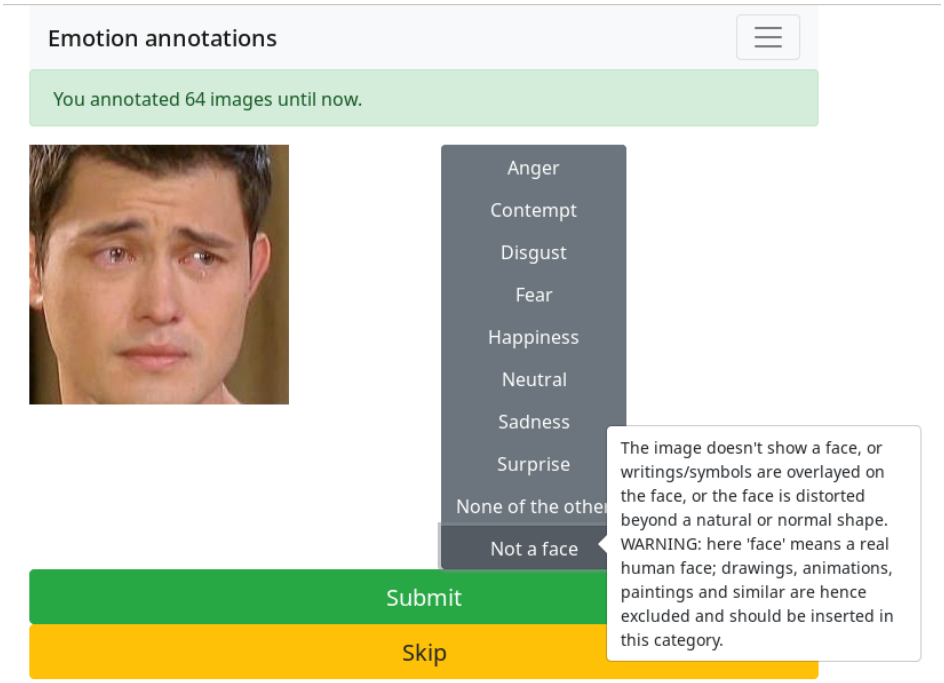


Figure 2.20: *Screenshot from the website used for gathering annotations*

Table 2.4: *Confusion matrix for the* no-face *class between AffectNet and WebAN*

		AffectNet	
		no-face	**face**
Crowd	**no-face**	31	4
	maybe	12	66
	face	7	430

considering that 50 images were used from each class as annotated in the original AffectNet dataset.

Network

With the acquired dataset, it could be possible to extract categorical labels from the various annotations and use them as ground truth for re-training the networks, replacing the new labels to the less reliable ones contained in the AffectNet dataset. Considering that simple majority vote is not appropriate when the reliability of the annotators is unknown and most importantly variable between them, various methods have been proposed to accomplish this [71, 72, 73]. However, a lot of information would be lost this way and the concern that ambiguous images could be interpreted in various ways would not be addressed. Here some approaches are described as possible proofs of concept of emotion classification done in a more "statistical" way, considering the

Table 2.5: *Absolute frequencies of the various classes in the WebAN dataset. The* Images *column indicates the number of images in which that emotion was the most annotated one.*

Emotion	Annotations	Images
Anger	445	32
Contempt	535	31
Disgust	462	30
Fear	461	28
Happiness	996	81
Neutral	1529	158
Sadness	663	45
Surprise	793	60
None	872	40
Total	6756	505

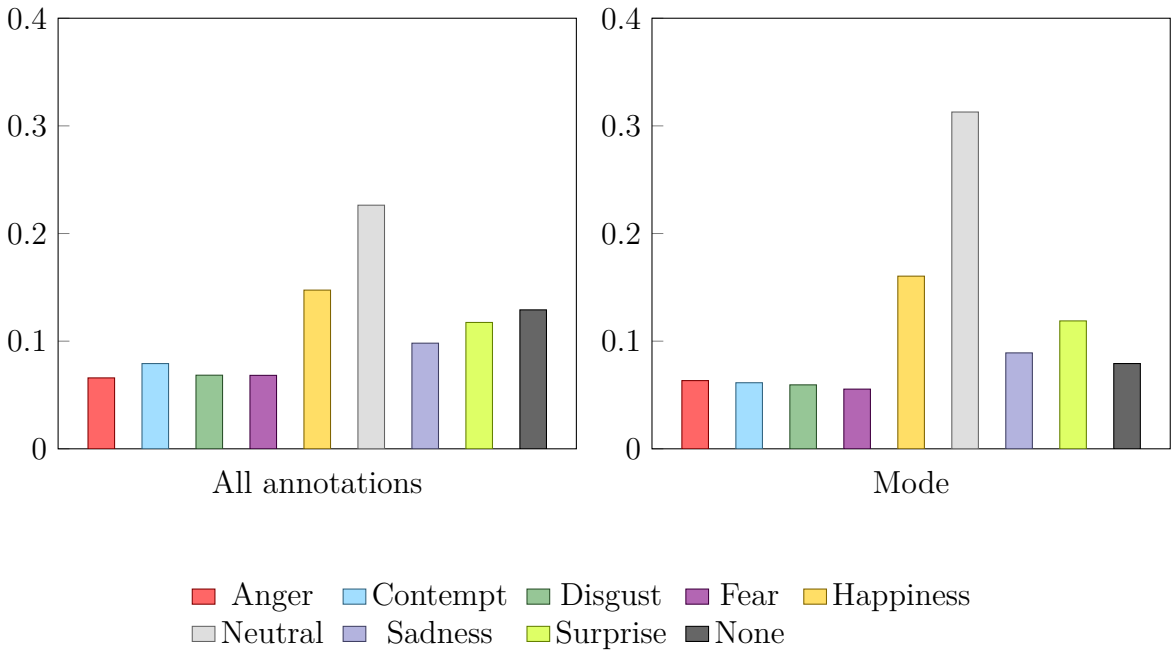


Figure 2.21: *Relative frequencies of the various classes in the WebAN dataset. The bar chart on the left takes into account all the gathered annotations, while the other one only counts the mode for each image, i.e. the class that was most labeled for each image.*

classes not as completely distinct categories but as fuzzy sets forming a partition. So, each image does not simply belong to a simple class, but for each class c a *membership function* μ_c can be defined such that:

$$\mu_c(i) \in [0,1] \quad \forall i \in \text{Images in the dataset}$$

and

$$\sum_{c=1}^{C} \mu_c(i) = 1 \quad \forall i \in \text{Images in the dataset}$$

The gathered dataset can then be used to estimate a sampling of those membership functions:

$$\mu_c(i) \approx \frac{\text{number of times image } i \text{ was labeled as } c}{\text{number of annotations for image } i}$$

Here, two methods are proposed for training neural networks that can actually learn the membership functions and one that performs a traditional classification but still exploits this *fuzzy ground truth*.

Regression model The first, simple approach is to try to learn the membership function values as any continuous function. The first part of the network is identical in structure to the networks used for classification, since the image processing necessary to extract important emotion-related features is the same. Only the last layer changes. Instead of using a *softmax* activation function which tries to keep only a single output value high while squashing the others to 0, the following activation function (aptly named *SumToOne*) should be used in order to have a generic output which only constraint is that the sum of all the values of the output is 1 and no value is below 0 (since it should estimate a fuzzy partition):

$$\text{SumToOne}([a_1, \ldots, a_N]^T) = \frac{[|a_1|, \ldots, |a_N|]^T}{\sum_{i=1}^{N} |a_i|}$$

The *mean squared error* (MSE) loss function can be used as the most common for regression tasks.

Geometric model The problem of simply using the mean squared error as loss function is that it does not take into account the meaning of the output and thus it is not capable of correctly assessing the error. For example, let's consider the following pairs of ground truths and predictions, respectively denoted by $\mathbf{T}$ and $\mathbf{P}$:

$$\mathbf{T} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{P} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{MSE}(\mathbf{T}, \mathbf{P}) = \frac{1}{18} \quad \mathbf{T} = \begin{bmatrix} \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{P} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \end{bmatrix} \text{MSE}(\mathbf{T}, \mathbf{P}) = \frac{1}{18}$$

The error is the same in both cases, but it is evident that the latter pair shows a much worse prediction. In the first case there is a specific correct label and the prediction shows an ambiguity split in half between the correct label and another one, in the second case there are multiple equivalent labels but the prediction ignores all of them and misclassifies the image to belong only to wrong classes.

A different loss function is thus proposed, described in Section D.1, which generalizes the idea of categorical cross entropy to fuzzy labels using a geometrical analogous of this problem. It was called Fuzzy Categorical Cross Entropy and defined as

$$\mathcal{F} = -2 \log \left(\sum_{c=1}^{C} \sqrt{t_c p_c} + \epsilon \right)$$

where t_c and p_c are the target and predicted value of the membership function for class c, C is the number of classes and ϵ is a small number used to prevent the logarithm from exploding. As a practical example, $\epsilon = 10^{-8}$ can be chosen and the *SumToOne* activation function selected.

Categorical model In this case the same architecture of the original networks can be used since the task is actually the same, only with different ground truth data. The loss function has to be changed since the available targets are not categorical and the $\mathcal{F}*$ loss function, also described in Section D.1, can be selected:

$$\mathcal{F}^* = -2 \log \left(\sum_{c=1}^{C} \sqrt{t_c^* p_c} \right) + \alpha \left(1 - \sum_{c=1}^{C} p_c^2 \right)$$

where $t_c^* = \frac{t_c}{\max\limits_c t_c}$. The *Softmax* activation function can be used in this case, so that no ϵ is necessary as in the geometrical model. However, the α parameter has to be selected and it is suggested to choose:

$$\alpha = \frac{2C \log C}{C - 1} \approx 5 \qquad (\text{C} = 9)$$

so that the loss would evaluate to 0 for completely uncertain predictions to completely uncertain targets.

In all three cases, the observed agreement for the annotations of each image can be used to weight the importance of the examples during training, so that the networks would learn more from images in which clear emotions were present.

2.3 Experimental protocol

2.3.1 Evaluation of networks trained on AffectNet

The networks were evaluated on three validation sets:

1. the AffectNet validation set with some missing images, as described in Section 2.1.1
2. the dataset extracted from the Extended Cohn-Kanade Dataset, as described in Section 2.1.1
3. the WebAN dataset using the original AffectNet labels, as described in Section 2.2.5

For each trained network, a prediction was performed for each of the samples in the validation set and compared with the ground truths given by the AffectNet dataset, computing the relative confusion matrix efficiently in the GPU by creating a custom Keras metric function. Once the confusion matrices were built, some metrics were evaluated (described in Section D.2.1):

- per-class and global precision
- per-class and global sensitivity
- per-class and global specificity
- per-class and global F_1 score
- per-class and global accuracy

These metrics were analysed both to see how they changed among the various networks and if some classes consistently scored higher or lower in all networks, showing if some emotions are easier or harder to recognize. In the latter case, since many data points are available, a paired Student's t-test was performed for each pair of emotions.

Let $m_c(n)$ be the value of a metric m computed on the predictions of network n relative to class c. For each pair of classes c_1, c_2, let $\delta m_{c_1 c_2}$ be a random variable of which the following sampling is available:

$$\delta m_{c_1 c_2}(n) = m_{c_1}(n) - m_{c_2}(n)$$

Then, a t-test can be performed with the following hypotheses:

$$H_0 : \mathbb{E}(\delta m_{c_1 c_2}) = 0; \quad H_1 : \mathbb{E}(\delta m_{c_1 c_2}) > 0$$

The resulting pvalue can be used to determine if a certain class shows statistically significant higher values for the metric relative to another class, that is:

$$\mathbb{E}(m_{c_1}) > \mathbb{E}(m_{c_2})$$

2.3.2 Evaluation of networks trained on CK+

The same evaluations described above in Section 2.3.1 were performed on the networks that were trained on the CK+ dataset in order to see how using such simplified training data (only frontal face images showing posed expressions) affected the results. Since few networks were trained on this dataset, there were not enough data points to perform the Student's t-tests in this case.

2.3.3 Computational cost of training

Training neural networks has a high computational cost which varies greatly based upon the architecture and training parameters used. Both time of training and memory used can be considered.

During training, a timestamp was recorded at the end of each epoch. This was not done from the start, so data is available only for some of the networks. In addition, some networks had to be retrained multiple times due to failures during training (mostly memory allocation errors which prevented the training from continuing or being recovered), but only the final training data is reported.

Since the computer on which the network were trained was also used for development, the training would occasionally slow down due to a higher computational load on the machine during an epoch. Additionally, the training of the networks had to be paused sometimes, so a long time could pass between two timestamps. These events, though, were quite rare. In order to ignore those outliers, the *per epoch* time reported in Table 3.40 is the median of the recorded epoch durations for each network. The

data for the first 50 and last 50 epochs is reported separately since the training parameters changed in a lot of networks and thus the median training time changed too. The total training time is also based on the median epoch duration. Because of this, the reported training times are actually shorter than the actual training times that were needed, but they are more in line with the time that would have been needed if the training was done on a dedicated machine.

As for the memory usage, an estimation was computed by taking into account the output shapes of each layer, the number of trainable parameters and the batch size used for training:

$$M = F\left(B\sum_{l=0}^{L} N_l + \sum_{l=0}^{L} P_l\right)$$

where M is the estimated total memory usage, F is the size (in bytes) of a single floating point number stored in memory, B is the batch size used for training, L is the number of layers in the network, N_l is the number of output neurons in a specific layer, P_l is the number of parameters of a specific layer. This is a lower estimate as it does not take into account other sources of memory usage such as variables needed by the optimizer and intermediate computations. During use, the memory usage drops dramatically due to the much lower batch size (generally it is fixed to 1, but it can be raised to increase the speed when multiple faces are present at the same time).

2.3.4 Annotators agreement

Some measures can be extracted from the dataset described in Section 2.2.5 in order to evaluate how hard this task is for humans and distinguish good examples from ambiguous ones.

Using the metrics described in Section D.3, in particular the annotators agreement as computed from the coincidence matrix of the annotations and the related κ index, it is possible to assess how much humans agree among each other both on the whole dataset and on single images[27] and this can be interpreted as an estimate of the accuracy of a hypothetical human annotator assuming the existence of real "true labels". When the agreement is very low, it is arguable that such a label does not exist and instead the shown expression should be considered as a combination of the various emotions (even if the subject of the picture intended it to be a single emotion, here only what is actually *expressed* can be evaluated).

The same agreement metrics can be applied (slightly modified as described in Section D.3.3) to the outputs of the neural networks. It can then be tested if the net-

[27] When computing the Fleiss' kappa for single images, the expected agreement of the whole dataset is still used

works recognize ambiguities by comparing these per-image agreements with the ones from the annotators. The Pearson correlation coefficient is computed between the two agreement values and then a Student t-test is performed on it with a null hypothesis of uncorrelation. The test is performed both on a per-network basis and on the *average network* by using the average agreement metrics.

The various annotators can also be compared to the AffectNet annotators using the latter as hypothetical ground truth. This can be done by taking into account only the modal label for each image or, as described in Section D.2.2, it is possible to consider all the annotations as if they were done by a single annotator. This way, the same metrics used to evaluate the neural networks can be computed for this *average human*. In order to make the resulting data useful for comparison with the neural networks, only annotations featuring one of the eight classes of the training set were considered.

2.3.5 Networks re-evaluation on WebAN dataset

The dataset described in Section 2.2.5 can be used to re-evaluate the networks with the new data. In particular, the evaluation focused on the classification accuracy and variations of it. Instead of the classical accuracy (which checks that both ground truth and prediction present a maximum for the same class) a pseudo-accuracy can be used, defined as:

$$\widehat{\text{accuracy}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\mu_{p_i}(i)}{\max_c \mu_c(i)}$$

where N is the number of images in the dataset, p_i is the prediction of the network for the ith image (expressed as the class corresponding to the maximum output), $\mu_c(i)$ is the ground truth value for class c relative to image i. Using this modified accuracy each prediction is evaluated based on how many human annotators made the same choice, getting a score of 1 if the most annotated class is predicted down to 0 if no annotator chose the same class (like for classical accuracy), with all the values in between.

A different improvement is to compute the accuracy as a weighted average were the weights are observed agreement of the annotations for each image, this way errors (or correct answers) are more significant for images that show clear emotions rather than ones that are ambiguous to humans too:

$$\text{accuracy}^w = \frac{1}{\sum_{i=1}^{N} P_i}\sum_{i=1}^{N} P_i \begin{cases} 1 & \text{if } p_i = \text{argmax}_c \mu_c(i) \\ 0 & \text{otherwise} \end{cases}$$

$$\widehat{\text{accuracy}}^w = \frac{1}{\sum_{i=1}^{N} P_i}\sum_{i=1}^{N} P_i \frac{\mu_{p_i}(i)}{\max_c \mu_c(i)}$$

Note that Fleiss' kappa is not suitable as a weight because it can have negative values.

Table 2.6: *Summary of the experimental protocol*

Evaluation of networks trained on AffectNet	
Network evaluation	
Metrics	Accuracy, Precision, Sensitivity, Specificity, F_1 score
Classes	Anger, Contempt, Disgust, Fear, Happiness, Neutral, Sadness, Surprise, Global
Datasets	AffectNet, CK+, WebAN
Student t-test (m: metric; c_1, c_2: classes)	
$\mathbf{H_0}$	$\mathbb{E}(m_{c_1}) = \mathbb{E}(m_{c_2})$
$\mathbf{H_1}$	$\mathbb{E}(m_{c_1}) > \mathbb{E}(m_{c_2})$
Evaluation of networks trained on CK+	
Network evaluation	
Metrics	Accuracy, Precision, Sensitivity, Specificity, F_1 score
Classes	Anger, Contempt, Disgust, Fear, Happiness, Neutral, Sadness, Surprise, Global
Datasets	AffectNet, CK+, WebAN
Computational cost of training	
Training time	Median epoch duration (epochs 1 through 50 and 51 through 100), expected total time
Allocated memory	Approximate memory usage from network structure and batch size
Annotators agreement	
Agreement metrics	Observed agreement ($\overline{P}$), Fleiss' kappa (κ)
Student t-test	
$\mathbf{H_0}$	no correlation between annotators agreement and network agreement
$\mathbf{H_1}$	correlation between annotators agreement and network agreement
Annotators against AffectNet	
Metrics	Accuracy, Precision, Sensitivity, Specificity, F_1 score
Classes	Anger, Contempt, Disgust, Fear, Happiness, Neutral, Sadness, Surprise, Global
Labelling	All annotations, most annotated label only
Networks re-evaluation on WebAN dataset	
Network evaluation	
Metrics	Accuracy, pseudo-accuracy
Classes	Global
Weight	Weighted on annotators agreement, unweighted

Chapter 3

Results

3.1 Evaluation of networks trained on AffectNet

3.1.1 AffectNet

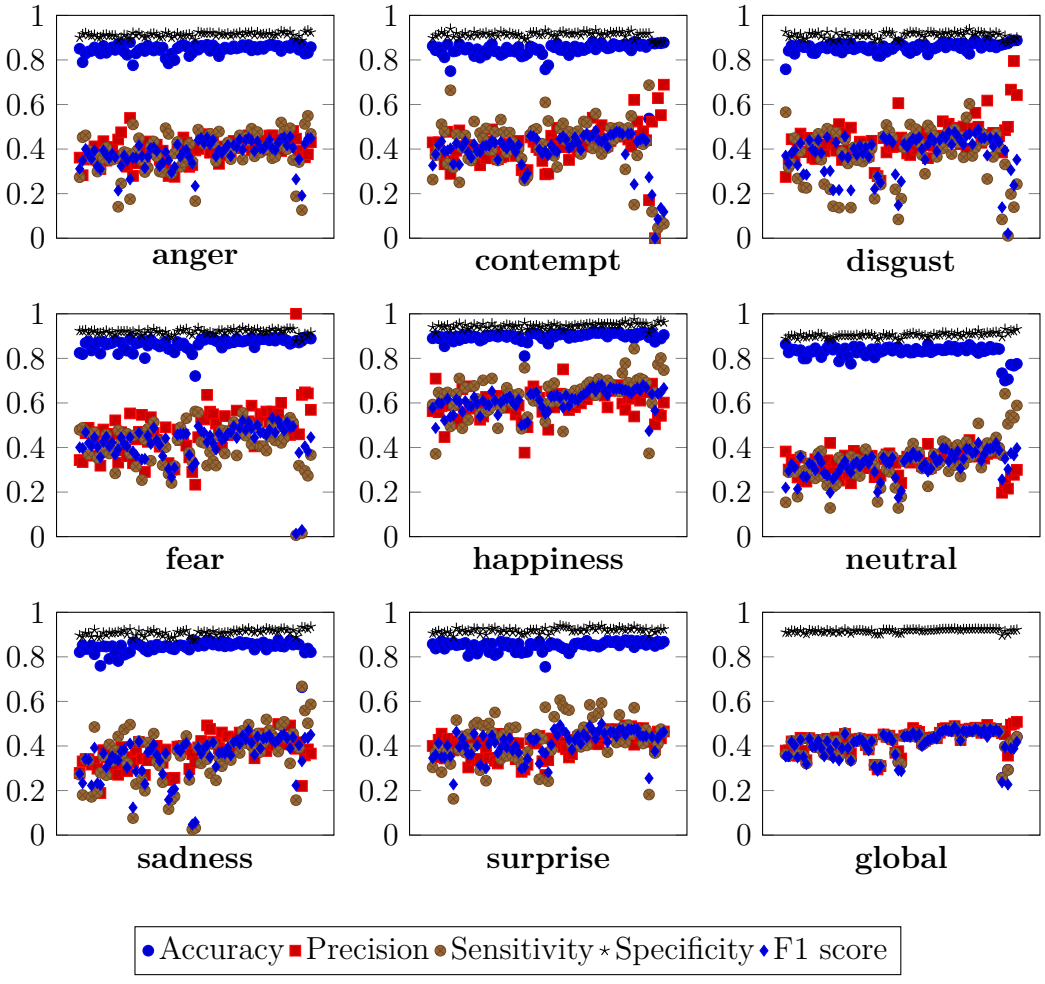


Figure 3.1: *Evaluation on the AffectNet dataset*

In Figure 3.1 the result of the evaluation of the various metrics is reported. For each class and metric, the various marks represent the different networks.

Those values are summarized in Tables 3.1 to 3.9 for each class and metric by reporting the minimum, median and maximum value in addition to the mean and standard deviation among all the networks.

Figure 3.2 shows the pvalues of the t-tests for the various metrics.

Table 3.1: *Summary metrics on the AffectNet dataset for class "anger"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.776	0.851	0.879	0.847	0.020
Precision	0.275	0.396	0.539	0.394	0.052
Sensitivity	0.126	0.388	0.549	0.384	0.085
Specificity	0.887	0.913	0.931	0.913	0.009
F1 score	0.190	0.393	0.456	0.380	0.055

Table 3.2: *Summary metrics on the AffectNet dataset for class "contempt"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.537	0.851	0.880	0.844	0.042
Precision	0.0	0.417	0.688	0.415	0.091
Sensitivity	0.0	0.427	0.686	0.409	0.125
Specificity	0.873	0.917	0.939	0.914	0.012
F1 score	0.0	0.415	0.482	0.391	0.089

Table 3.3: *Summary metrics on the AffectNet dataset for class "disgust"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.758	0.859	0.887	0.856	0.019
Precision	0.257	0.428	0.795	0.438	0.082
Sensitivity	0.010	0.402	0.603	0.367	0.123
Specificity	0.875	0.915	0.937	0.911	0.013
F1 score	0.020	0.411	0.487	0.379	0.090

Table 3.4: *Summary metrics on the AffectNet dataset for class "fear"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.720	0.872	0.895	0.865	0.027
Precision	0.233	0.473	1.0	0.474	0.107
Sensitivity	0.006	0.431	0.561	0.411	0.098
Specificity	0.878	0.921	0.936	0.919	0.010
F1 score	0.012	0.443	0.531	0.422	0.084

Table 3.5: *Summary metrics on the AffectNet dataset for class "happiness"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.810	0.900	0.920	0.898	0.015
Precision	0.376	0.609	0.750	0.602	0.060
Sensitivity	0.371	0.640	0.843	0.634	0.083
Specificity	0.914	0.947	0.975	0.946	0.010
F1 score	0.474	0.618	0.682	0.612	0.048

Table 3.6: *Summary metrics on the AffectNet dataset for class "neutral"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.700	0.835	0.868	0.828	0.030
Precision	0.197	0.345	0.433	0.333	0.049
Sensitivity	0.128	0.339	0.641	0.338	0.094
Specificity	0.883	0.904	0.932	0.904	0.009
F1 score	0.175	0.334	0.420	0.327	0.053

Table 3.7: *Summary metrics on the AffectNet dataset for class "sadness"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.663	0.849	0.874	0.843	0.028
Precision	0.188	0.384	0.498	0.374	0.064
Sensitivity	0.025	0.372	0.667	0.355	0.121
Specificity	0.877	0.909	0.935	0.908	0.013
F1 score	0.047	0.375	0.469	0.352	0.089

Table 3.8: *Summary metrics on the AffectNet dataset for class "surprise"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.755	0.855	0.874	0.850	0.020
Precision	0.269	0.421	0.492	0.407	0.050
Sensitivity	0.162	0.443	0.605	0.430	0.093
Specificity	0.890	0.920	0.940	0.919	0.010
F1 score	0.227	0.429	0.5	0.412	0.057

Table 3.9: *Summary metrics on the AffectNet dataset for class "global"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.257	0.430	0.487	0.416	0.051
Precision	0.294	0.435	0.508	0.430	0.046
Sensitivity	0.255	0.429	0.487	0.416	0.051
Specificity	0.895	0.918	0.926	0.917	0.007
F1 score	0.227	0.423	0.484	0.409	0.057

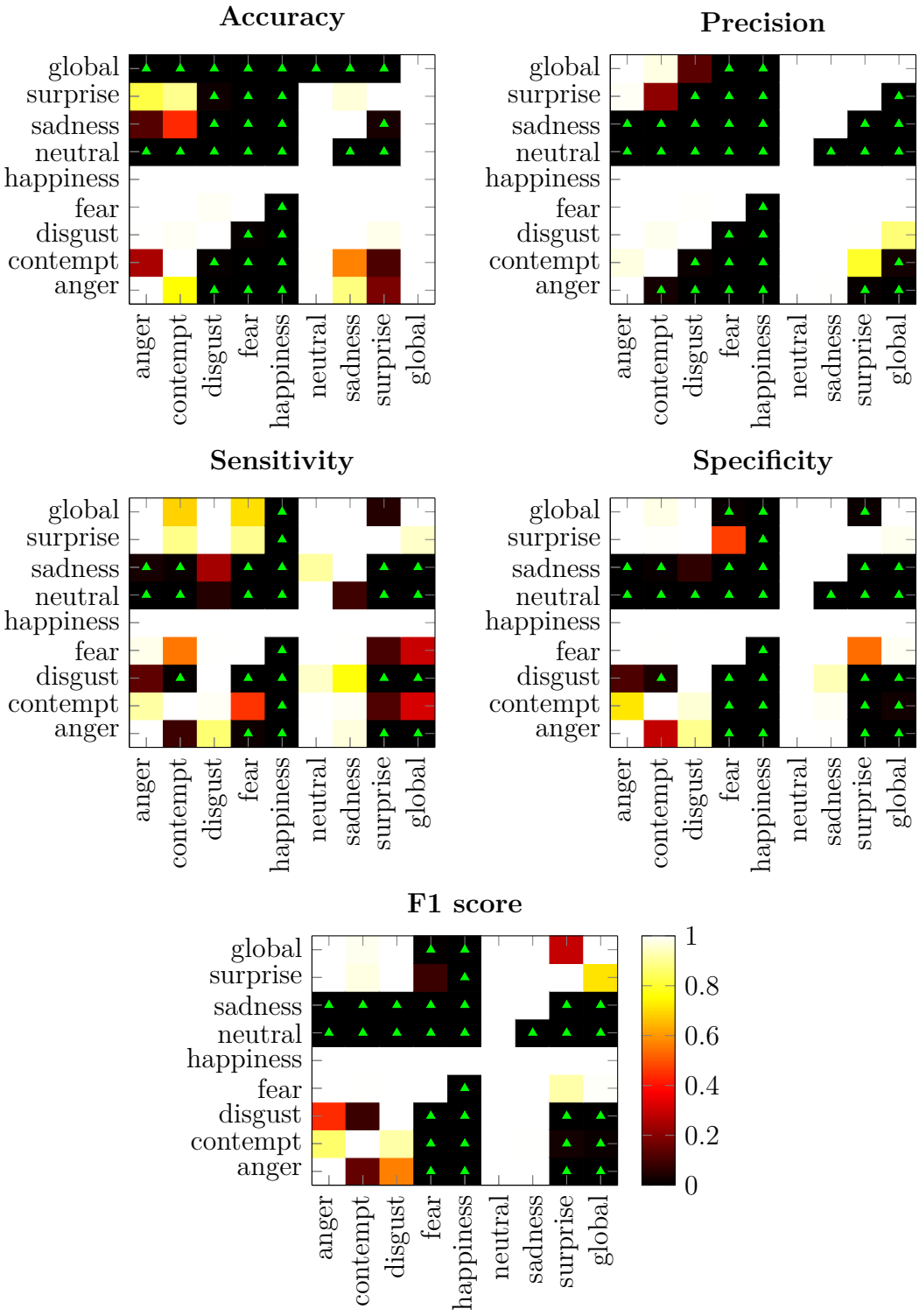


Figure 3.2: *Result of the Student's t-tests on the AffectNet dataset. Each cell shows the pvalue for the test with alternative hypothesis "the expected value of the metric for the class on the horizontal axis is higher than for the one on the vertical axis". A green triangle indicates that the null hypothesis was rejected* ($\text{pvalue} < 0.05$) *and hence the metric evaluates to statistically higher values for the class on the horizontal axis.*

3.1.2 Extended Cohn-Kanade

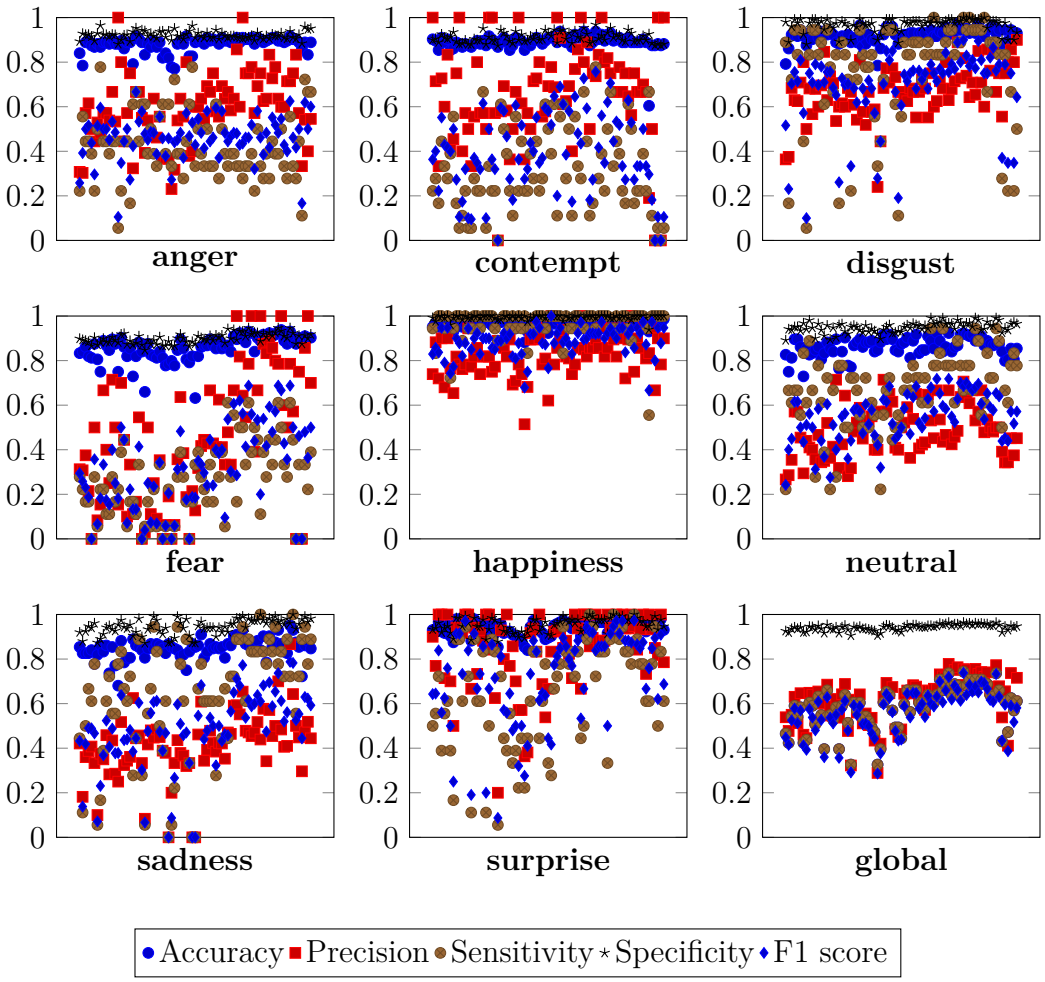


Figure 3.3: *Evaluation on the CK+ dataset*

In Figure 3.3 the result of the evaluation of the various metrics is reported. For each class and metric, the various marks represent the different networks.

Those values are summarized in Tables 3.10 to 3.18 for each class and metric by reporting the minimum, median and maximum value in addition to the mean and standard deviation among all the networks.

Figure 3.4 shows the pvalues of the t-tests for the various metrics.

Table 3.10: *Summary metrics on the CK+ dataset for class "anger"*

	Min	Median	Max	μ	σ
Accuracy	0.770	0.888	0.916	0.880	0.032
Precision	0.230	0.6	1.0	0.593	0.154
Sensitivity	0.055	0.388	0.777	0.423	0.146
Specificity	0.881	0.917	0.965	0.920	0.016
F1 score	0.105	0.468	0.666	0.463	0.096

Table 3.11: *Summary metrics on the CK+ dataset for class "contempt"*

	Min	Median	Max	μ	σ
Accuracy	0.604	0.888	0.937	0.889	0.037
Precision	0.0	0.666	1.0	0.655	0.224
Sensitivity	0.0	0.277	0.777	0.317	0.191
Specificity	0.874	0.905	0.968	0.909	0.022
F1 score	0.0	0.384	0.756	0.392	0.183

Table 3.12: *Summary metrics on the CK+ dataset for class "disgust"*

	Min	Median	Max	μ	σ
Accuracy	0.784	0.923	0.972	0.920	0.032
Precision	0.24	0.68	0.9	0.670	0.116
Sensitivity	0.055	0.833	1.0	0.758	0.236
Specificity	0.880	0.975	1.0	0.966	0.030
F1 score	0.1	0.75	0.9	0.686	0.170

Table 3.13: *Summary metrics on the CK+ dataset for class "fear"*

	Min	Median	Max	μ	σ
Accuracy	0.631	0.868	0.937	0.855	0.057
Precision	0.0	0.416	1.0	0.445	0.296
Sensitivity	0.0	0.277	0.611	0.260	0.165
Specificity	0.846	0.899	0.946	0.899	0.022
F1 score	0.0	0.312	0.687	0.312	0.192

Table 3.14: *Summary metrics on the CK+ dataset for class "happiness"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.881	0.972	1.0	0.968	0.019
Precision	0.514	0.818	1.0	0.828	0.091
Sensitivity	0.555	1.0	1.0	0.962	0.068
Specificity	0.939	1.0	1.0	0.994	0.009
F1 score	0.666	0.9	1.0	0.886	0.064

Table 3.15: *Summary metrics on the CK+ dataset for class "neutral"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.729	0.868	0.923	0.862	0.041
Precision	0.266	0.481	0.714	0.487	0.113
Sensitivity	0.222	0.666	0.944	0.662	0.166
Specificity	0.891	0.951	0.991	0.949	0.022
F1 score	0.242	0.571	0.717	0.546	0.107

Table 3.16: *Summary metrics on the CK+ dataset for class "sadness"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.680	0.854	0.951	0.847	0.045
Precision	0.0	0.423	0.866	0.412	0.146
Sensitivity	0.0	0.666	1.0	0.627	0.264
Specificity	0.871	0.948	1.0	0.945	0.034
F1 score	0.0	0.489	0.787	0.483	0.177

Table 3.17: *Summary metrics on the CK+ dataset for class "surprise"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.840	0.951	0.993	0.943	0.036
Precision	0.2	0.9	1.0	0.849	0.165
Sensitivity	0.055	0.722	1.0	0.660	0.238
Specificity	0.877	0.961	1.0	0.953	0.031
F1 score	0.086	0.8	0.972	0.722	0.207

Table 3.18: *Summary metrics on the CK+ dataset for class "global"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.326	0.604	0.743	0.584	0.096
Precision	0.289	0.645	0.778	0.617	0.107
Sensitivity	0.326	0.604	0.743	0.584	0.096
Specificity	0.905	0.944	0.963	0.942	0.013
F1 score	0.286	0.586	0.737	0.561	0.101

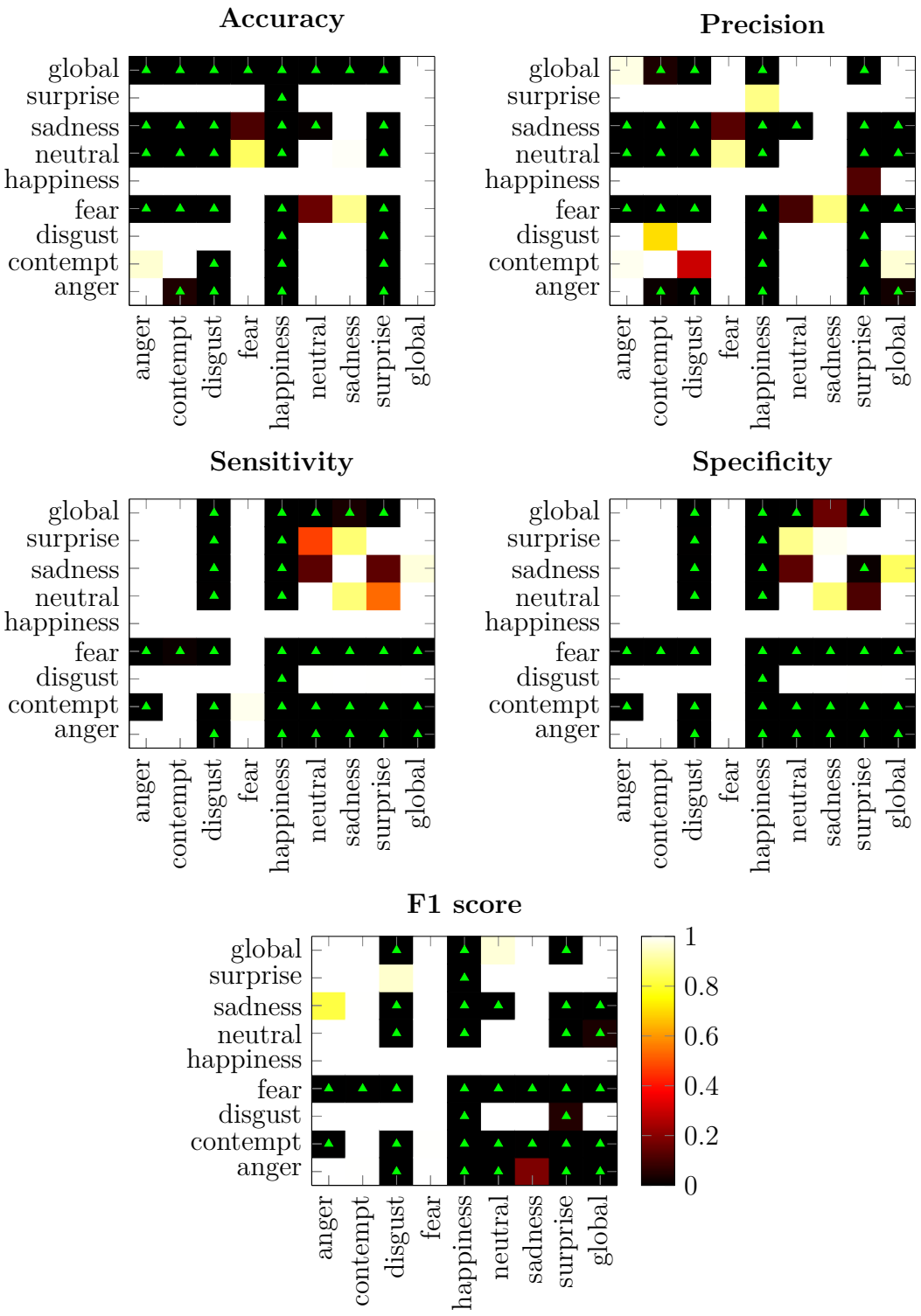


Figure 3.4: *Result of the Student's t-tests on the CK+ dataset. Each cell shows the pvalue for the test with alternative hypothesis "the expected value of the metric for the class on the horizontal axis is higher than for the one on the vertical axis". A green triangle indicates that the null hypothesis was rejected (*$\text{pvalue} < 0.05$*) and hence the metric evaluates to statistically higher values for the class on the horizontal axis.*

3.1.3 WebAN dataset with AffectNet labels

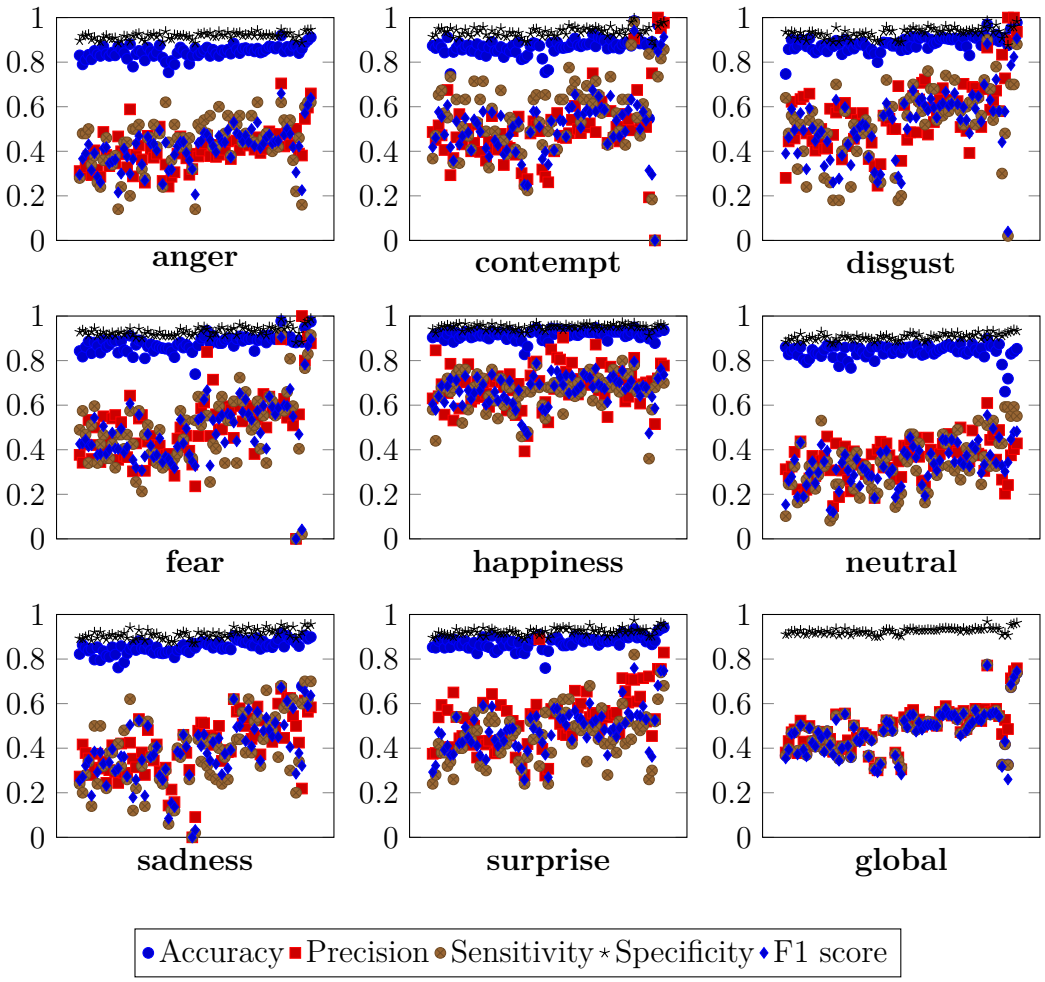


Figure 3.5: *Evaluation on the WebAN dataset*

In Figure 3.5 the result of the evaluation of the various metrics is reported. For each class and metric, the various marks represent the different networks.

Those values are summarized in Tables 3.19 to 3.27 for each class and metric by reporting the minimum, median and maximum value in addition to the mean and standard deviation among all the networks.

Figure 3.6 shows the pvalues of the t-tests for the various metrics.

Table 3.19: *Summary metrics on the WebAN dataset for class "anger"*

	Min	Median	Max	μ	σ
Accuracy	0.754	0.848	0.918	0.845	0.027
Precision	0.241	0.404	0.704	0.409	0.085
Sensitivity	0.14	0.46	0.62	0.436	0.114
Specificity	0.885	0.919	0.945	0.917	0.014
F1 score	0.205	0.431	0.659	0.413	0.087

Table 3.20: *Summary metrics on the WebAN dataset for class "contempt"*

	Min	Median	Max	μ	σ
Accuracy	0.546	0.873	0.984	0.867	0.053
Precision	0.0	0.486	1.0	0.494	0.157
Sensitivity	0.0	0.571	0.979	0.556	0.161
Specificity	0.875	0.938	0.997	0.936	0.020
F1 score	0.0	0.525	0.941	0.511	0.140

Table 3.21: *Summary metrics on the WebAN dataset for class "disgust"*

	Min	Median	Max	μ	σ
Accuracy	0.746	0.886	0.977	0.884	0.034
Precision	0.245	0.571	1.0	0.568	0.148
Sensitivity	0.02	0.56	0.9	0.521	0.171
Specificity	0.875	0.935	0.985	0.931	0.022
F1 score	0.039	0.551	0.907	0.524	0.147

Table 3.22: *Summary metrics on the WebAN dataset for class "fear"*

	Min	Median	Max	μ	σ
Accuracy	0.739	0.881	0.977	0.878	0.036
Precision	0.0	0.5	1.0	0.507	0.158
Sensitivity	0.0	0.489	0.914	0.485	0.162
Specificity	0.881	0.929	0.988	0.931	0.019
F1 score	0.0	0.469	0.905	0.481	0.147

Table 3.23: *Summary metrics on the WebAN dataset for class "happiness"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.827	0.918	0.949	0.916	0.020
Precision	0.392	0.695	0.903	0.687	0.092
Sensitivity	0.36	0.68	0.8	0.662	0.080
Specificity	0.913	0.953	0.970	0.951	0.010
F1 score	0.470	0.681	0.795	0.669	0.067

Table 3.24: *Summary metrics on the WebAN dataset for class "neutral"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.660	0.843	0.898	0.838	0.034
Precision	0.147	0.368	0.609	0.354	0.086
Sensitivity	0.081	0.326	0.591	0.332	0.119
Specificity	0.878	0.906	0.938	0.906	0.013
F1 score	0.120	0.336	0.555	0.332	0.088

Table 3.25: *Summary metrics on the WebAN dataset for class "sadness"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.678	0.853	0.916	0.851	0.034
Precision	0.0	0.415	0.666	0.405	0.129
Sensitivity	0.0	0.38	0.7	0.380	0.161
Specificity	0.872	0.910	0.955	0.911	0.019
F1 score	0.0	0.377	0.673	0.382	0.137

Table 3.26: *Summary metrics on the WebAN dataset for class "surprise"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.759	0.881	0.941	0.875	0.028
Precision	0.277	0.538	0.888	0.529	0.122
Sensitivity	0.24	0.46	0.82	0.459	0.117
Specificity	0.892	0.923	0.973	0.923	0.015
F1 score	0.258	0.482	0.759	0.481	0.099

Table 3.27: *Summary metrics on the WebAN dataset for class "global"*

	Min	**Median**	**Max**	μ	σ
Accuracy	0.306	0.491	0.774	0.479	0.093
Precision	0.300	0.500	0.770	0.494	0.091
Sensitivity	0.306	0.490	0.775	0.479	0.093
Specificity	0.901	0.927	0.968	0.926	0.013
F1 score	0.260	0.483	0.771	0.474	0.098

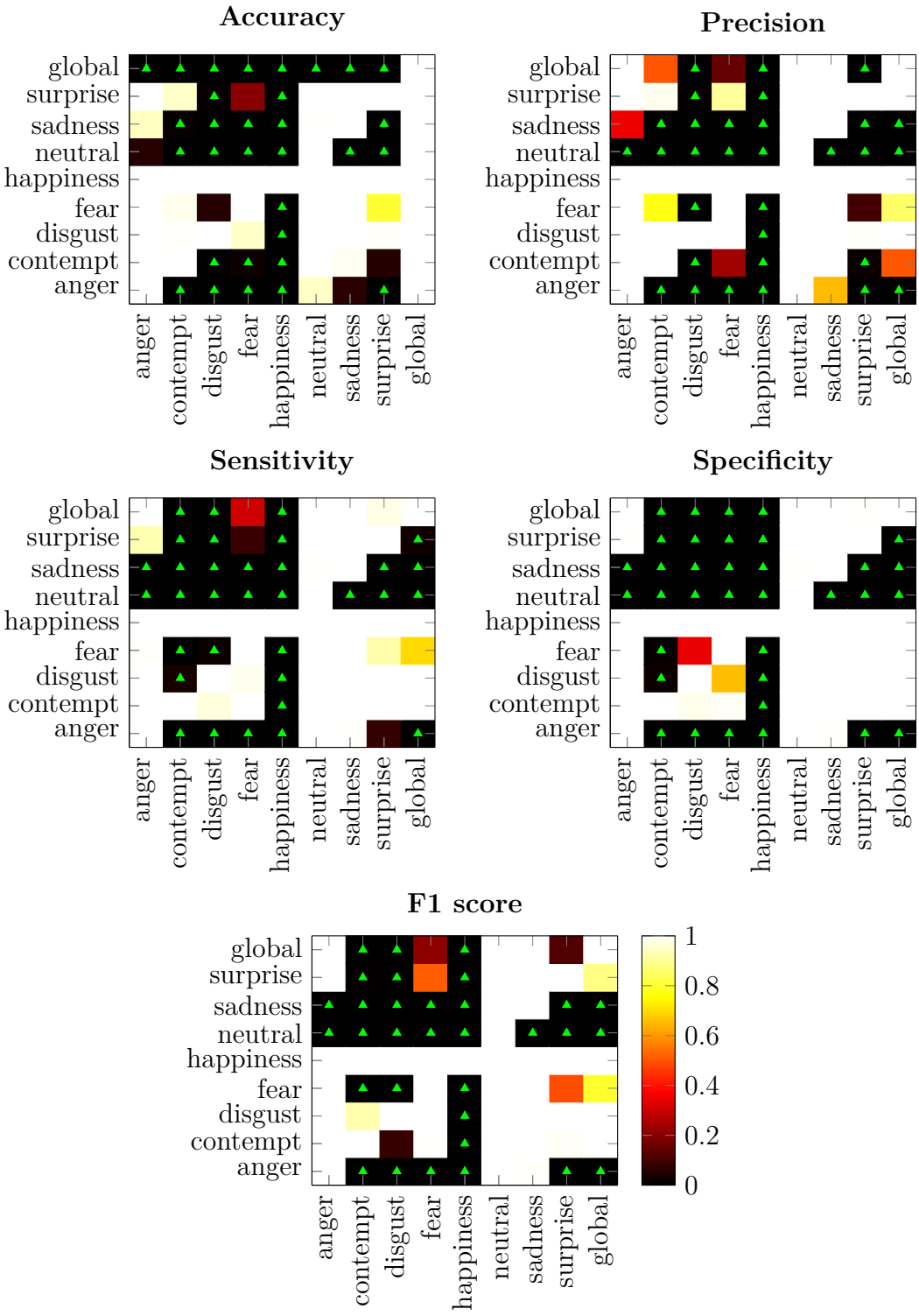


Figure 3.6: *Result of the Student's t-tests on the WebAN dataset. Each cell shows the pvalue for the test with alternative hypothesis "the expected value of the metric for the class on the horizontal axis is higher than for the one on the vertical axis". A green triangle indicates that the null hypothesis was rejected* ($\text{pvalue} < 0.05$) *and hence the metric evaluates to statistically higher values for the class on the horizontal axis.*

3.2 Evaluation of networks trained on CK+

Tables 3.28 to 3.37 show the result of the evaluation of the various metrics for the networks that were trained on the CK+ dataset.

Each network was trained three times reaching slightly different results, but only the mean value of the metrics for those three training sessions is reported.

Table 3.28: *Evaluation result for network `Compare1` on the AffectNet dataset*

Network:	`Compare1`		**Dataset:**	AffectNet
Emotion	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1 score**
anger	0.779	0.193	0.202	0.884	0.179
contempt	0.751	0.174	0.260	0.884	0.207
disgust	0.841	0.258	0.138	0.884	0.179
fear	0.785	0.135	0.138	0.879	0.129
happiness	0.838	0.354	0.333	0.904	0.342
neutral	0.815	0.183	0.143	0.881	0.150
sadness	0.812	0.173	0.123	0.879	0.136
surprise	0.780	0.213	0.275	0.892	0.239
global	0.202	0.210	0.202	0.886	0.195

Table 3.29: *Evaluation result for network `Compare2` on the AffectNet dataset*

Network:	`Compare2`		**Dataset:**	AffectNet
Emotion	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1 score**
anger	0.776	0.212	0.277	0.892	0.235
contempt	0.754	0.170	0.240	0.882	0.199
disgust	0.837	0.275	0.178	0.887	0.211
fear	0.817	0.170	0.130	0.883	0.142
happiness	0.861	0.476	0.434	0.918	0.443
neutral	0.777	0.197	0.248	0.887	0.219
sadness	0.848	0.224	0.079	0.879	0.114
surprise	0.792	0.227	0.272	0.894	0.244
global	0.233	0.244	0.232	0.890	0.226

Table 3.30: *Evaluation result for network `Compare3` on the AffectNet dataset*

Network:	`Compare3`		**Dataset:**	AffectNet
Emotion	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1 score**
anger	0.124	0.124	1.0	0.0	0.220
contempt	0.873	0.0	0.0	0.873	0.0
disgust	0.874	0.0	0.0	0.874	0.0
fear	0.877	0.0	0.0	0.877	0.0
happiness	0.873	0.0	0.0	0.873	0.0
neutral	0.874	0.0	0.0	0.874	0.0
sadness	0.874	0.0	0.0	0.874	0.0
surprise	0.876	0.0	0.0	0.876	0.0
global	0.124	0.015	0.125	0.765	0.027

Table 3.31: *Evaluation result for network `Compare4` on the AffectNet dataset*

Network:	`Compare4`		**Dataset:**	AffectNet
Emotion	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1 score**
anger	0.857	0.160	0.027	0.876	0.043
contempt	0.331	0.123	0.702	0.865	0.210
disgust	0.841	0.136	0.052	0.875	0.072
fear	0.863	0.154	0.027	0.878	0.043
happiness	0.856	0.303	0.079	0.878	0.116
neutral	0.862	0.127	0.016	0.874	0.028
sadness	0.837	0.091	0.034	0.873	0.048
surprise	0.813	0.139	0.098	0.877	0.115
global	0.131	0.154	0.129	0.875	0.084

Table 3.32: *Evaluation result for network `Compare1` on the CK+ dataset*

Network: `Compare1`			**Dataset:** CK+		
Emotion	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1 score**
anger	0.976	0.896	0.925	0.989	0.909
contempt	0.990	0.962	0.962	0.994	0.962
disgust	0.988	0.947	0.962	0.994	0.954
fear	0.981	0.942	0.907	0.987	0.919
happiness	0.990	0.950	0.981	0.997	0.964
neutral	0.937	0.811	0.592	0.945	0.669
sadness	0.960	0.823	0.944	0.991	0.871
surprise	0.988	0.929	0.981	0.997	0.954
global	0.907	0.908	0.907	0.987	0.900

Table 3.33: *Evaluation result for network `Compare2` on the CK+ dataset*

Network: `Compare2`			**Dataset:** CK+		
Emotion	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1 score**
anger	0.993	0.966	0.981	0.997	0.972
contempt	0.995	0.964	1.0	1.0	0.981
disgust	1.0	1.0	1.0	1.0	1.0
fear	1.0	1.0	1.0	1.0	1.0
happiness	1.0	1.0	1.0	1.0	1.0
neutral	0.981	0.981	0.870	0.981	0.920
sadness	1.0	1.0	1.0	1.0	1.0
surprise	0.993	0.947	1.0	1.0	0.972
global	0.981	0.982	0.981	0.997	0.981

Table 3.34: *Evaluation result for network `Compare3` on the CK+ dataset*

Network:	`Compare3`		**Dataset:**	CK+
Emotion	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1 score**
anger	0.125	0.125	1.0	0.0	0.222
contempt	0.875	0.0	0.0	0.875	0.0
disgust	0.875	0.0	0.0	0.875	0.0
fear	0.875	0.0	0.0	0.875	0.0
happiness	0.875	0.0	0.0	0.875	0.0
neutral	0.875	0.0	0.0	0.875	0.0
sadness	0.875	0.0	0.0	0.875	0.0
surprise	0.875	0.0	0.0	0.875	0.0
global	0.125	0.015	0.125	0.765	0.027

Table 3.35: *Evaluation result for network `Compare4` on the CK+ dataset*

Network:	`Compare4`		**Dataset:**	CK+
Emotion	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1 score**
anger	0.856	0.398	0.259	0.899	0.301
contempt	0.740	0.290	0.740	0.952	0.417
disgust	0.884	0.549	0.518	0.932	0.519
fear	0.898	0.763	0.259	0.903	0.381
happiness	0.918	0.659	0.740	0.962	0.694
neutral	0.865	0.442	0.222	0.896	0.284
sadness	0.842	0.378	0.370	0.911	0.349
surprise	0.909	0.653	0.555	0.938	0.596
global	0.458	0.517	0.458	0.924	0.443

Table 3.36: *Evaluation result for network `Compare1` on the WebAN dataset*

Network:	`Compare1`		**Dataset:**	WebAN
Emotion	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1 score**
anger	0.778	0.200	0.226	0.885	0.193
contempt	0.770	0.209	0.292	0.893	0.242
disgust	0.850	0.298	0.133	0.883	0.184
fear	0.785	0.136	0.163	0.885	0.144
happiness	0.840	0.381	0.42	0.914	0.398
neutral	0.812	0.148	0.115	0.879	0.116
sadness	0.816	0.175	0.14	0.880	0.151
surprise	0.788	0.232	0.28	0.891	0.253
global	0.221	0.222	0.221	0.889	0.210

Table 3.37: *Evaluation result for network `Compare2` on the WebAN dataset*

Network:	`Compare2`		**Dataset:**	WebAN
Emotion	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1 score**
anger	0.789	0.242	0.293	0.893	0.261
contempt	0.750	0.162	0.244	0.884	0.194
disgust	0.850	0.341	0.186	0.889	0.233
fear	0.815	0.112	0.078	0.880	0.089
happiness	0.860	0.490	0.480	0.924	0.465
neutral	0.795	0.245	0.299	0.897	0.268
sadness	0.843	0.203	0.086	0.878	0.120
surprise	0.794	0.262	0.32	0.897	0.283
global	0.249	0.257	0.248	0.893	0.239

Table 3.38: *Evaluation result for network `Compare3` on the WebAN dataset*

Network:	`Compare3`		**Dataset:**	WebAN
Emotion	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1 score**
anger	0.126	0.126	1.0	0.0	0.224
contempt	0.875	0.0	0.0	0.875	0.0
disgust	0.873	0.0	0.0	0.873	0.0
fear	0.881	0.0	0.0	0.881	0.0
happiness	0.873	0.0	0.0	0.873	0.0
neutral	0.875	0.0	0.0	0.875	0.0
sadness	0.873	0.0	0.0	0.873	0.0
surprise	0.873	0.0	0.0	0.873	0.0
global	0.126	0.015	0.125	0.765	0.028

Table 3.39: *Evaluation result for network `Compare4` on the WebAN dataset*

Network:	`Compare4`		**Dataset:**	WebAN
Emotion	**Accuracy**	**Precision**	**Sensitivity**	**Specificity**	**F1 score**
anger	0.857	0.041	0.013	0.872	0.020
contempt	0.326	0.124	0.748	0.896	0.212
disgust	0.852	0.274	0.08	0.878	0.117
fear	0.867	0.314	0.035	0.882	0.059
happiness	0.844	0.176	0.08	0.877	0.101
neutral	0.863	0.060	0.013	0.875	0.022
sadness	0.843	0.194	0.073	0.876	0.105
surprise	0.823	0.144	0.073	0.874	0.095
global	0.139	0.166	0.139	0.879	0.091

3.3 Computational cost of training

Table 3.40 reports the time that was needed to train the various networks. Since all networks were trained three times, three rows are present for each network. The *per epoch* training time is reported separately for the first 50 and the last 50 epochs because in most cases the batch size was different in the second half of the training.

Table 3.41 reports an estimation of the memory usage during training (computed with the formula described in Section 2.3.3). As explained above, the training parameters often changed between the first half and second half of training, needing different amounts of memory. It was chosen to report only the maximum of the two values,

Table 3.40: *Time needed to train the various networks, both as the median time for training epoch and as a total.*

Net	Training time			Net	Training time		
	Per epoch		*Total*		*Per epoch*		*Total*
	1-50	51-100			1-50	51-100	
	1m 32s	1m 8s	2h 14m		7.942s	2m 18s	2h 2m
`test11`	1m 9s	1m 15s	2h 0m	`test22`	7.955s	2m 6s	1h 52m
	1m 9s	1m 15s	2h 0m		8.004s	2m 4s	1h 50m
	1m 1s	56.97s	1h 38m		8.272s	1m 17s	1h 11m
`test13`	1m 1s	55.81s	1h 37m	`test23`	4.995s	1m 18s	1h 9m
	56.29s	59.98s	1h 36m		4.957s	1m 18s	1h 9m
	3.596s	0s	2m 59s		10m 24s	10m 24s	17h 20m
`test14`	3.592s	0s	2m 59s	`test24`	10m 31s	10m 30s	17h 31m
	3.592s	0s	2m 59s		10m 27s	10m 27s	17h 25m
	1m 8s	0s	57m 11s		10m 29s	10m 29s	17h 28m
`test15`	1m 19s	0s	1h 6m	`test25`	10m 31s	10m 28s	17h 29m
	57.18s	0s	47m 39s		10m 30s	10m 29s	17h 30m
	5.092s	1m 19s	1h 10m		5m 31s	5m 31s	9h 12m
`test16`	5.092s	1m 19s	1h 10m	`test26`	5m 35s	5m 35s	9h 19m
	5.077s	1m 19s	1h 10m		5m 34s	5m 34s	9h 17m
	6.014s	1m 33s	1h 23m		5m 35s	5m 34s	9h 18m
`test17`	7.298s	1m 33s	1h 24m	`test27`	5m 37s	5m 37s	9h 23m
	5.992s	1m 33s	1h 22m		5m 34s	5m 35s	9h 18m
	4.906s	1m 16s	1h 7m		8.451s	8.419s	14m 3s
`test18`	4.889s	1m 16s	1h 7m	`Compare1`	8.498s	8.451s	14m 7s
	4.904s	1m 16s	1h 7m		8.466s	8.513s	14m 9s
	5.769s	1m 30s	1h 19m		9.981s	10.08s	16m 43s
`test19`	5.779s	1m 32s	1h 22m	`Compare2`	10.49s	10.46s	17m 28s
	8.438s	1m 33s	1h 24m		10.37s	10.49s	17m 23s
	10.20s	1m 43s	1h 34m		11.98s	12.14s	20m 6s
`test20`	6.564s	1m 44s	1h 32m	`Compare3`	9.755s	11.01s	17m 18s
	6.680s	1m 44s	1h 32m		9.990s	9.910s	16m 35s
	8.431s	2m 11s	1h 56m		8.991s	9.003s	14m 59s
`test21`	8.455s	2m 12s	1h 57m	`Compare4`	8.991s	8.987s	14m 58s
	8.414s	2m 13s	1h 58m		9.021s	9.037s	15m 2s

since that is the actual amount of memory needed to train the network.

Table 3.41: *Approximate memory needed to train the various networks*

Net	Memory	Net	Memory	Net	Memory
	2.3 GiB		147.3 MiB		1.9 GiB
test0	4.5 GiB	test11	287.6 MiB	test22	1.9 GiB
	367.6 MiB		18.5 MiB		1.9 GiB
	2.1 GiB		239.2 MiB		1.9 GiB
test1	4 GiB	test12	464.3 MiB	test23	1.9 GiB
	338.3 MiB		32.5 MiB		1.9 GiB
	1.5 GiB		91.4 MiB		414.2 MiB
test2	3 GiB	test13	91.4 MiB	test24	414.2 MiB
	245.2 MiB		91.4 MiB		414.2 MiB
	1.4 GiB		1.1 GiB		414.2 MiB
test3	2.7 GiB	test14	1.1 GiB	test25	414.2 MiB
	225.6 MiB		1.1 GiB		414.2 MiB
	2.1 GiB		23.5 MiB		296 MiB
test4	4 GiB	test15	23.5 MiB	test26	296 MiB
	338.3 MiB		23.5 MiB		296 MiB
	1.4 GiB		2.1 GiB		296 MiB
test5	2.7 GiB	test16	2.1 GiB	test27	296 MiB
	225.6 MiB		2.1 GiB		296 MiB
	594.8 MiB		1.5 GiB		4.7 MiB
test6	1.1 GiB	test17	1.5 GiB	Compare1	4.7 MiB
	91.4 MiB		1.5 GiB		4.7 MiB
	149.2 MiB		1.1 GiB		18.8 MiB
test7	287.2 MiB	test18	1.1 GiB	Compare2	18.8 MiB
	22.6 MiB		1.1 GiB		18.8 MiB
	271.6 MiB		2 GiB		28.2 MiB
test8	271.6 MiB	test19	2 GiB	Compare3	28.2 MiB
	271.6 MiB		2 GiB		28.2 MiB
	181.1 MiB		3.2 GiB		5.4 MiB
test9	181.1 MiB	test20	3.2 GiB	Compare4	5.4 MiB
	181.1 MiB		3.2 GiB		5.4 MiB
	587 MiB		2.7 GiB		
test10	1.1 GiB	test21	2.7 GiB		
	74.1 MiB		2.7 GiB		

3.4 Annotators agreement

Figure 3.7 shows the observed agreement ($\bar{P}$) and the corresponding Fleiss' kappa (κ) for each image in the dataset.

When the same metrics were computed on the output of the various networks (only for the images which were labeled as one of the eight base emotions in the original AffectNet dataset) and correlation with the above results was tested, all the tests rejected the null hypothesis of uncorrelation (except for the one on `test26`, which

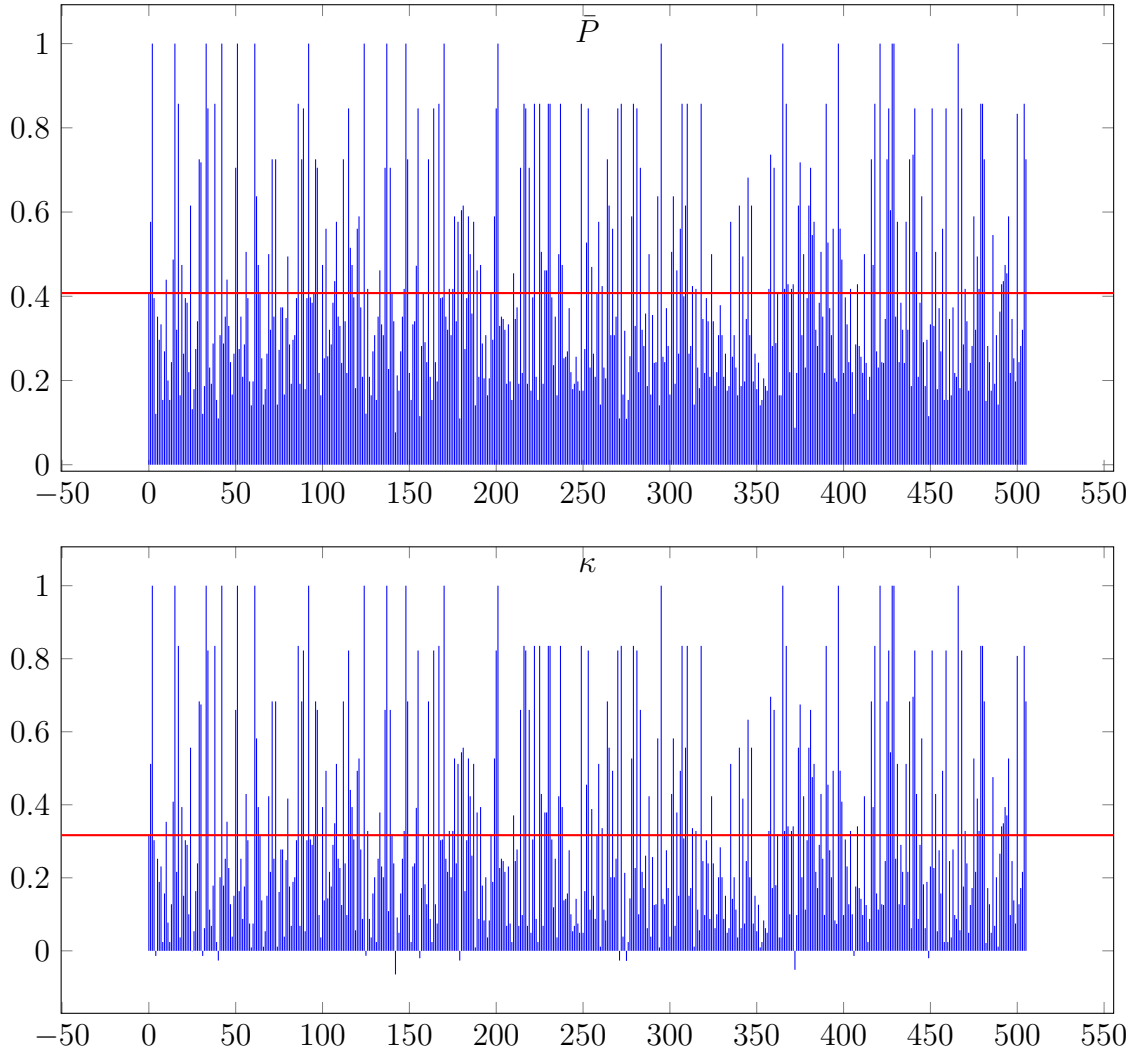


Figure 3.7: *Observed agreement and Fleiss' kappa of the gathered annotations. Each bar represents an image in the WebAN dataset and the red lines show the value of the two metrics evaluated on all annotations together.*

showed strong overfitting behaviour during training) with pvalue $< 2 \cdot 10^{-5}$.

Table 3.42 shows the result of the evaluation of the gathered annotations compared against the AffectNet labels.

When the same comparison is made by only considering the most annotated labels for each image, the results in Table 3.43 are obtained.

Table 3.42: *Evaluation of the annotators against the AffectNet labels*

Emotion	Accuracy	Precision	Sensitivity	Specificity	F_1 score
Anger	0.868	0.431	0.286	0.906	0.344
Contempt	0.823	0.161	0.136	0.892	0.147
Disgust	0.880	0.568	0.370	0.909	0.448
Fear	0.873	0.5	0.353	0.909	0.414
Happiness	0.901	0.614	0.721	0.955	0.663
Neutral	0.794	0.323	0.617	0.938	0.425
Sadness	0.864	0.476	0.433	0.916	0.454
Surprise	0.853	0.411	0.472	0.925	0.439
Global	0.429	0.435	0.423	0.919	0.417

Table 3.43: *Evaluation of the most annotated labels against the AffectNet labels*

Emotion	Accuracy	Precision	Sensitivity	Specificity	F_1 score
Anger	0.873	0.531	0.309	0.901	0.390
Contempt	0.813	0.117	0.076	0.875	0.093
Disgust	0.897	0.724	0.375	0.910	0.494
Fear	0.909	0.666	0.382	0.926	0.486
Happiness	0.904	0.590	0.75	0.963	0.661
Neutral	0.775	0.330	0.823	0.969	0.471
Sadness	0.899	0.595	0.549	0.938	0.571
Surprise	0.861	0.473	0.490	0.922	0.482
Global	0.467	0.503	0.469	0.925	0.456

3.5 Networks re-evaluation on WebAN dataset

Figure 3.8 shows the various accuracy metrics evaluated for each network (horizontal axis) on the WebAN dataset.

The above result is summarized in the Table 3.44.

Table 3.44: *Summary metrics on the WebAN dataset using the gathered annotations*

	Unweighted		Weighted	
	Accuracy	Pseudo-accuracy	Accuracy	Pseudo-accuracy
Min	0.253	0.378	0.192	0.290
Median	0.379	0.515	0.292	0.403
Max	0.465	0.614	0.370	0.484
μ	0.374	0.513	0.286	0.396
σ	0.052	0.060	0.043	0.049

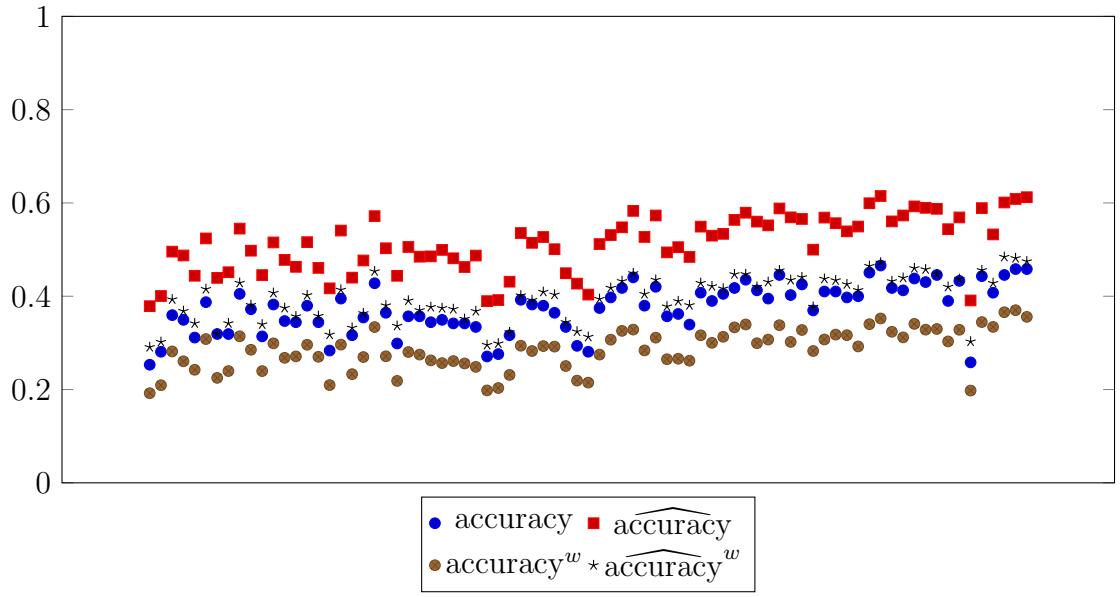


Figure 3.8: *Evaluation on the WebAN dataset using the gathered annotations*

Table 3.45: *Summary of the main results*

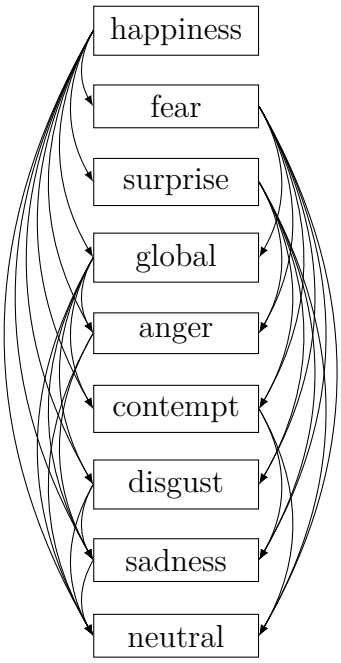
Evaluation of networks trained on AffectNet			
Network evaluation			
Best	**Global accuracy**		
network	*on AffectNet*	*on CK+*	*on WebAN*
`test20`	**48.7**%	68.75%	55.18%
`test21`	47.03%	**74.3**%	55.69%
`test24`	46.02%	68.75%	**77.46**%
Student t-test			
See to the right for detected priority of emotions based on F_1 score An arrow indicates that the score is generally lower for the pointed class			

Evaluation of networks trained on CK+			
Network evaluation			
Best	**Global accuracy**		
network	*on AffectNet*	*on CK+*	*on WebAN*
`Compare2`	**23.97**%	**98.61**%	**25.31**%

Computational cost of training		
Network	**Training time**	**Used memory**
`test0`	-	**4.5 GiB**
`test20`	1h 34m	3.2 GiB
`test24`	**17h 31m**	414.2 MiB

Annotators agreement			
	Minimum	**Average**	**Maximum**
Agreement	7.69%	40.67%	100%
κ	−0.064	0.316	1
Student t-test			
All passed except for `test26`			
Annotators against AffectNet			
	All annotations	**Majority vote**	
Global accuracy	43%	46.78%	

Networks re-evaluation on WebAN dataset
Network evaluation
All decreased except for pseudo-accuracy



Chapter 4

Discussion

4.1 Main findings

Considering both the global accuracy and the averaged F_1 score evaluated on the AffectNet validation set, the network that reached the best performance was `test20` which results are summarized in Table 4.1. The current state of the art [22] has an accuracy as high as 68% for the AffectNet dataset which is higher than the one reached by `test20`, but it must be noted that the best-performing networks are much deeper than the ones that could be considered in this work due to resource limits. Empirical tests on the real-time application showed that the network could recognize clear emotions quite well, struggling mostly in those cases where even humans could not be sure about the expressed emotion. It also did this without noticeable lag (except for the one due to the webcam itself), even when multiple people were present at the same time. Interestingly, `test20` is not one of the most complex networks that were trained, showing that a bigger network not always gives better results and it is important to properly tune the parameters before increasing the size. Examining the parameters of this network, it has the largest number of filters in the first CMR block among the trained networks together with small pool size and stride. Neither the second CMR block nor the fully connected layer show anything unique compared to the other networks, though. This suggests that in order to have a good classification, a good

Table 4.1: *Performance of `test20`*

Network name: `test20`		
Dataset	**Accuracy**	**F_1 score**
AffectNet	48.7%	48.45%
CK+	68.75%	68.09%
WebAN	55.18%	55.0%

selection of low-level features is paramount.

The other trained networks reached different levels of performance, with the best ones correctly classifying almost half of the images in the AffectNet validation set. This result becomes much more respectable when the performance of the networks and of real humans is compared on the same dataset. Evaluating the humans annotators on the WebAN dataset using the original AffectNet labels, they reached an accuracy of 42.9% when considered singularly and 46.7% when their annotations were combined by majority vote. The trained networks, instead, reached accuracies as high as 77.1% on the same dataset, with a median value of 47.4%. Thus, most of the trained neural networks outperformed the average human in this task.

The agreement metrics show how the low accuracies of the annotators are not due to the fact that the available labels were simply wrong, instead most images showed ambiguous emotion expressions, so that a definitive answer could not be given and the various annotators could not agree on the same label. The κ values even show that some images were so disputed that even by selecting their label at random the annotators would have agreed more. This goes to say that forcing a single label for describing an image is not only unsuitable but even misleading to the real understanding of what is expressed in it. The tests of correlation between the "agreement" of the predictions and that of the annotators show how the neural networks implicitly also learn which images are more ambiguous, even if they were never instructed to do so and no related data was given them.

Globally the networks also showed that some emotions are easier to recognize than others, with similar patterns for all metrics in the cross-tabulated t-tests, and this result is confirmed by the per-class metrics (particularly, the F_1 score can be used as a reference) related to the performance of the human annotators. In particular the *happiness* class (which is probably the most important for humans, as interestingly its recognition is relatively preserved even in Alzheimer disease patients [74]) was by far the easiest to recognize for both automatic classifier and human annotators

The networks reached the highest mean accuracy on the Extended Cohn-Kanade dataset, where only frontal face pictures of posed, often exaggerated, emotions were proposed and thus the labeling was much more univocal. The fact that the accuracy is not that high in this dataset too is probably linked to the fact that the networks were trained on the AffectNet dataset which, due to the above-mentioned ambiguity problems, prevented them from properly learning the stereotypical emotions shown in the CK+ dataset. The fact that the task was simpler still allowed the networks to perform better anyway. Conversely some of very simple networks that were trained on the CK+ dataset reached high accuracies on that same dataset, but performed very

poorly on the AffectNet dataset. This shows that such a simple dataset is not suitable to obtain an adequate generalization power.

The accuracies of the networks computed on the labeling done by the multiple annotators is generally lower than those related to the original labels. This is to be expected after looking at the previous results since the agreement of the gathered annotations with the AffectNet labels was pretty low, which means that the two labellings were quite discordant. The only metric that is higher is the *pseudo-accuracy* because it takes into account the classifications that were *almost* right, but it should not be compared with the accuracy computed on the AffectNet labels because in that case there was no information about alternative answers and their relative accuracy score. The fact that the weighted accuracy was lower than the standard accuracy indicates that while, as stated above, the networks learned how to distinguish clear from ambiguous images, they did not perform better on images showing more distinct emotions. It would however be desirable to have an higher accuracy on clearer images since an error in ambiguous images is usually considered less of a problem. This shows the importance of adding this weight during training in order to teach the networks mainly on the more representative images.

4.2 Technical challenges

Such a complex task as automatic emotion recognition requires huge amounts of computational power and time.

Unfortunately the available resources were not particularly powerful. A laptop such as the one used is not comparable to the clusters used to train state of the art networks, but by using the parallel processing capabilities of the GPU it was possible to train the neural networks in reasonable times, even if the network depth had to be kept quite low. The longest training took more than seventeen hours, but the real times that were necessary to train the networks were much longer. In fact, sometimes the training failed due to crashes or other problems in the computer and it had to be manually resumed or, worse, restarted from the beginning. In addition, there was a need to face the problem of memory size. For each individual network architecture, the maximal batch size had to be found by trial and error in order not to incur in memory overflow. The formula described in Section 2.3.3 could be used to predict if a network would fit in the GPU memory, but it was not accurate enough to prevent memory allocation errors. Sometimes, the training would start but then an error would occur later on due to the less-than-optimal memory management of Keras. When this happened, all the previous training time was wasted and new parameters had to be found.

State-of-the-art networks are quite deep, as reported in Section 1.3, but in this project it was not possible to use too many layers due to the above-mentioned memory constraints. This led the project to focus on the first layers and optimize them. As the results show, this is actually very important to get a powerful network, but it is often overlooked in favour of adding more and more layers.

Since this work explored the real-time applicability of FER, another concern was that the system could be too slow to make predictions at a fast enough pace. Using various expedients such as using a lower image resolution for landmark detection than for actual face morphing during normalization and making the neural networks predict the emotions while the rest of the program was busy acquiring the next webcam image made it possible for the real-time application to actually run in real time.

4.3 Work limitations

The number of possible network architectures is infinite and in this work only a few of the simpler ones have been tested. Due to resource and time constraint it was not possible to build very complex networks nor to train them for long period of times, so the focus was kept on trying variations of the first few layers, exploring how the most low-level features affected the result and, conversely, if some invariants could be found. Some networks reached convergence while others could probably improve a bit if trained for some epoch more, but the data acquired was still useful to better understand the peculiarities of this task and find interesting behaviours that are visible from early on in the training process. It would still be useful to further explore the parameters space of those networks for a more comprehensive analysis.

Even after considering the above limitations, the amount of acquired data is huge and only a little part of it could be analyzed in this work due to time constraints and to avoid making the length of this document overwhelming. Chapter 3 reports various results from which more information than those discussed in Section 4.1 could be obtained, but also other information like training data (e.g. loss, accuracy, overfitting, etc.) and more low-level evaluation data than the metrics proposed have been set aside in order to focus on the reported results.

4.4 Expectation verification

With reference to the expectations in Section 1.2.2:

1. the developed classifiers did not reach state-of-the-art performance due to resource limitations, but they still managed to be more accurate than humans on

the AffectNet dataset (containing very general images of facial expressions)

2. the real-time application was developed and the classifiers could be used with it
3. the accuracy of the classifiers on the CK+ dataset (good illumination, frontal angle, posed expressions) was higher than in the more general AffectNet dataset
4. classifiers trained on the CK+ dataset reached high accuracy in the same dataset but were incapable of generalizing to less "standard" images
5. the classifiers showed significant differences in classification power for the various emotions (in particular, the *happiness* label was the most distinguishable of all)
6. annotations from humans showed low agreement for most proposed images, indicating that the emotion expression is often ambiguous
7. considering "almost right" answers improved the accuracy while taking into account the ambiguity decreased it, it would however be necessary to use the information during training rather than validation in order to get significant results

Chapter 5

Conclusions

In this project a real-time application that can use trained convolutional neural networks to recognize emotions from webcam images was successfully developed. While this work only includes an initial analysis of the task, it brings some interesting points to the discussion of how to best approach the problem. It proves the intuitive notion that the task of emotion recognition is very *hard*, for humans too, and traditional classifications approaches are probably not perfectly suited for it. The results suggest that a more probabilistic approach to labeling should be taken, maybe following the recent trend of expressing emotions as continuous values in the arousal-valence space. This latter approach, though, does not address the fact that humans do actually reason in a categorical way when thinking about emotions. The difference in complexity of recognizing the various emotion should also be taken into account when the detected expression is acted upon: an happy face can be detected with relative ease, but it is easy to mistake a face showing contempt for something else.

5.1 Future developments

As discussed in Section 4.3, there is still a lot of data obtained during the development of this work to be analyzed. Just as an example, Figure 5.1 shows an initial analysis of how the accuracy changes during training targeted at finding a way to predict its behaviour from the first epochs in order to train for a long time only the networks that look promising. Development of some basic methods and mathematical support for them was started but abandoned in favour of the presented results.

The networks proposed in Section 2.2.5 should also be implemented, tested and expanded.

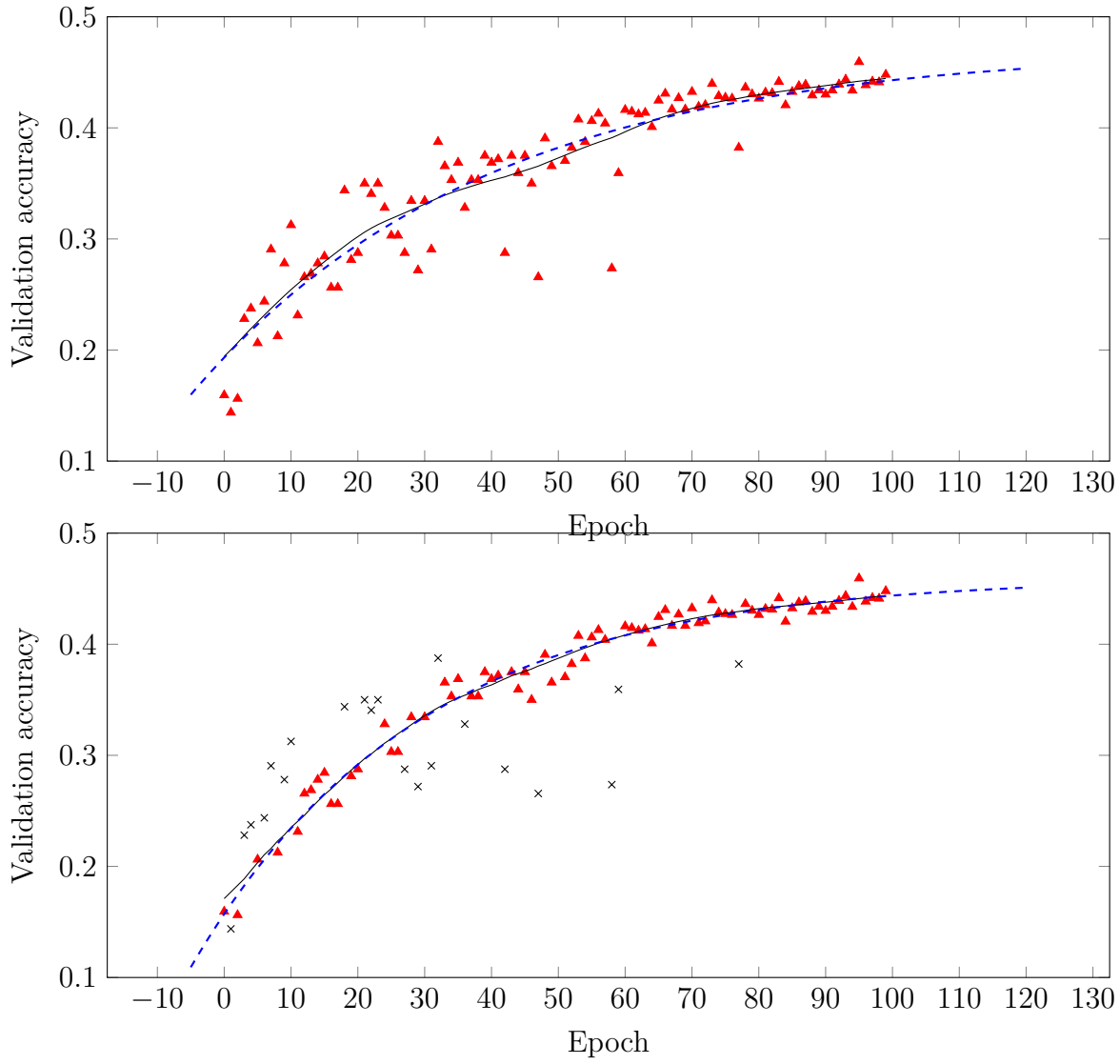


Figure 5.1: *Validation accuracy during training of a network. The image on the top shows the raw data while the one on the bottom was cleaned with a specifically developed automatic detection algorithm of outliers. The black line is obtained by averaging piece-wise linear approximations of the data points evaluated on a moving window, the blue line is an exponential approximation from which explicit parameters can be obtained: starting accuracy, maximum reachable accuracy and both start-of-training and end-of-training slope.*

Bibliography

[1] 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*, 32:478–487, June 2016.

[2] 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 : the official journal of the National Academy of Neuropsychologists*, 29:374–384, June 2014.

[3] Hanne Cecilie Braarud, Siv Skotheim, Kjartan Høie, Maria Wik Markhus, Marian Kjellevold, Ingvild Eide Graff, Jan Øystein Berle, and Kjell Morten Stormark. Affective facial expression in sub-clinically depressed and non-depressed mothers during contingent and non-contingent face-to-face interactions with their infants. *Infant behavior & development*, 48:98–104, August 2017.

[4] Marco Leo, Pierluigi Carcagnì, Cosimo Distante, Paolo Spagnolo, Pier Luigi Mazzeo, Anna Chiara Rosato, Serena Petrocchi, Chiara Pellegrino, Annalisa Levante, Filomena De Lumè, and Flavia Lecciso. Computational assessment of facial expression production in asd children. *Sensors (Basel, Switzerland)*, 18, November 2018.

[5] Umesh Thonse, Rishikesh V Behere, Samir Kumar Praharaj, and Podila Sathya Venkata Narasimha Sharma. Facial emotion recognition, socio-occupational functioning and expressed emotions in schizophrenia versus bipolar disorder. *Psychiatry research*, 264:354–360, June 2018.

[6] 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*, 86:42–54, April 2014.

[7] Duncan R Babbage, Jackki Yim, Barbra Zupan, Dawn Neumann, Machiko R Tomita, and Barry Willer. Meta-analysis of facial affect recognition difficulties after traumatic brain injury. *Neuropsychology*, 25:277–285, May 2011.

[8] Nicolas Hamelin, Othmane El Moujahid, and Park Thaichon. Emotion and advertising effectiveness: A novel facial expression analysis approach. *Journal of Retailing and Consumer Services*, 36:103–111, 2017.

[9] A. Mehrabian. *Nonverbal Communication*. Taylor & Francis, 2017.

[10] Zsófia Borsos and Miklos Gyori. Can automated facial expression analysis show differences between autism and typical functioning? *Studies in health technology and informatics*, 242:797–804, 2017.

[11] Joseph Manfredonia, Abigail Bangerter, Nikolay V Manyakov, Seth Ness, David Lewin, Andrew Skalkin, Matthew Boice, Matthew S Goodwin, Geraldine Dawson, Robert Hendren, Bennett Leventhal, Frederick Shic, and Gahan Pandina. Automatic recognition of posed facial expression of emotion in individuals with autism spectrum disorder. *Journal of autism and developmental disorders*, 49:279–293, January 2019.

[12] Mira Jeong and Byoung Chul Ko. Driver's facial expression recognition in real-time for safe driving. *Sensors (Basel, Switzerland)*, 18, December 2018.

[13] Mrs Ayesha Butalia, Dr. Maya Ingle, and Dr. Parag Kulkarni. Facial expression recognition for security.

[14] Bee Theng Lau. Portable real time emotion detection system for the disabled. 37:6561–6566, 2010.

[15] Rami Alazrai and C. S. George Lee. Real-time emotion identification for socially intelligent robots, 2012.

[16] Caifeng Shan, Shaogang Gong, and Peter W. McOwan. Facial expression recognition based on local binary patterns: A comprehensive study. *Image and Vision Computing*, 27(6):803 – 816, 2009.

[17] G. Zhao and M. Pietikainen. Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):915–928, June 2007.

[18] R. Zhi, M. Flierl, Q. Ruan, and W. B. Kleijn. Graph-preserving sparse nonnegative matrix factorization with application to facial expression recognition. *Part B (Cybernetics) IEEE Transactions on Systems, Man, and Cybernetics*, 41(1):38–52, February 2011.

[19] Shan Li and Weihong Deng. Deep facial expression recognition: A survey. *CoRR*, abs/1804.08348, 2018.

[20] P. Liu, S. Han, Z. Meng, and Y. Tong. Facial expression recognition via a boosted deep belief network. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1805–1812, June 2014.

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

[22] A. Mollahosseini, B. Hasani, and M. H. Mahoor. Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, page 1, 2018.

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

[24] H. Jung, S. Lee, J. Yim, S. Park, and J. Kim. Joint fine-tuning in deep neural networks for facial expression recognition. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 2983–2991, December 2015.

[25] S. He, S. Wang, W. Lan, H. Fu, and Q. Ji. Facial expression recognition using deep boltzmann machine from thermal infrared images. In *Proc. Humaine Association Conf. Affective Computing and Intelligent Interaction*, pages 239–244, September 2013.

[26] Zhan Wu, Tong Chen, Ying Chen, Zhihao Zhang, and Guangyuan Liu. Nirexpnet: Three-stream 3d convolutional neural network for near infrared facial expression recognition. *Applied Sciences*, 7:1184, 2017.

[27] Earnest Paul Ijjina and C. Krishna Mohan. Facial expression recognition using kinect depth sensor and convolutional neural networks, 2014.

[28] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, page I, December 2001.

[29] Nianyin Zeng, Hong Zhang, Baoye Song, Weibo Liu, Yurong Li, and Abdullah M Dobaie. Facial expression recognition via learning deep sparse autoencoders. *Neurocomputing*, 273:643–649, 2018.

[30] Kaihao Zhang, Yongzhen Huang, Yong Du, and Liang Wang. Facial expression recognition based on deep evolutional spatial-temporal networks. 26:4193–4203, 2017.

[31] Anbang Yao, Dongqi Cai, Ping Hu, Shandong Wang, Liang Sha, and Yurong Chen. Holonet: Towards robust emotion recognition in the wild. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, ICMI '16, pages 472–478, New York, NY, USA, 2016. ACM.

[32] Ping Hu, Dongqi Cai, Shandong Wang, Anbang Yao, and Yurong Chen. Learning supervised scoring ensemble for emotion recognition in the wild. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, ICMI '17, pages 553–560, New York, NY, USA, 2017. ACM.

[33] M. Shin, M. Kim, and D. Kwon. Baseline CNN structure analysis for facial expression recognition. In *Proc. 25th IEEE Int. Symp. Robot and Human Interactive Communication (RO-MAN)*, pages 724–729, August 2016.

[34] Sarah Adel Bargal, Emad Barsoum, Cristian Canton Ferrer, and Cha Zhang. Emotion recognition in the wild from videos using images. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, ICMI '16, pages 433–436, New York, NY, USA, 2016. ACM.

[35] Tong Zhang, Wenming Zheng, Zhen Cui, Yuan Zong, Jingwei Yan, and Keyu Yan. A deep neural network-driven feature learning method for multi-view facial expression recognition. 18:2528–2536, 2016.

[36] T. Devries, K. Biswaranjan, and G. W. Taylor. Multi-task learning of facial landmarks and expression. In *Proc. Canadian Conf. Computer and Robot Vision*, pages 98–103, May 2014.

[37] Z. Meng, P. Liu, J. Cai, S. Han, and Y. Tong. Identity-aware convolutional neural network for facial expression recognition. In *Proc. 12th IEEE Int. Conf. Automatic Face Gesture Recognition (FG 2017)*, pages 558–565, May 2017.

[38] C. Zhang, P. Wang, K. Chen, and J. Kämäräinen. Identity-aware convolutional neural networks for facial expression recognition. *Journal of Systems Engineering and Electronics*, 28(4):784–792, August 2017.

[39] Diah Anggraeni Pitaloka, Ajeng Wulandari, T. Basaruddin, and Dewi Yanti Liliana. Enhancing cnn with preprocessing stage in automatic emotion recognition. 116:523–529, 2017.

[40] André Teixeira Lopes, Edilson de Aguiar, Alberto F De Souza, and Thiago Oliveira-Santos. Facial expression recognition with convolutional neural networks: coping with few data and the training sample order. *Pattern Recognition*, 61:610–628, 2017.

[41] Yijun Gan. Facial expression recognition using convolutional neural network. In *Proceedings of the 2Nd International Conference on Vision, Image and Signal Processing*, ICVISP 2018, pages 29:1–29:5, New York, NY, USA, 2018. ACM.

[42] Samira Ebrahimi Kahou, Christopher Pal, Xavier Bouthillier, Pierre Froumenty, Çaglar Gülçehre, Roland Memisevic, Pascal Vincent, Aaron Courville, Yoshua Bengio, Raul Chandias Ferrari, Mehdi Mirza, Sébastien Jean, Pierre-Luc Carrier, Yann Dauphin, Nicolas Boulanger-Lewandowski, Abhishek Aggarwal, Jeremie Zumer, Pascal Lamblin, Jean-Philippe Raymond, Guillaume Desjardins, Razvan Pascanu, David Warde-Farley, Atousa Torabi, Arjun Sharma, Emmanuel Bengio, Myriam Côté, Kishore Reddy Konda, and Zhenzhou Wu. Combining modality specific deep neural networks for emotion recognition in video. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*, ICMI '13, pages 543–550, New York, NY, USA, 2013. ACM.

[43] Boris Knyazev, Roman Shvetsov, Natalia Efremova, and Artem Kuharenko. Convolutional neural networks pretrained on large face recognition datasets for emotion classification from video.

[44] A. Dapogny and K. Bailly. Investigating deep neural forests for facial expression recognition. In *Proc. 13th IEEE Int. Conf. Automatic Face Gesture Recognition (FG 2018)*, pages 629–633, May 2018.

[45] Veena Mayya, Radhika M Pai, and MM Manohara Pai. Automatic facial expression recognition using dcnn. *Procedia Computer Science*, 93:453–461, 2016.

[46] Ligang Zhang, Brijesh Verma, Dian Tjondronegoro, and Vinod Chandran. Facial expression analysis under partial occlusion: A survey. *ACM Comput. Surv.*, 51(2):25:1–25:49, April 2018.

[47] M. Liu, S. Li, S. Shan, and X. Chen. Au-aware deep networks for facial expression recognition. In *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, pages 1–6, April 2013.

[48] Hatice Gunes and Björn Schuller. Categorical and dimensional affect analysis in continuous input: Current trends and future directions. 31:120–136, 2013.

[49] S. Du, Y. Tao, and A. M. Martinez. Compound facial expressions of emotion. 111:E1454–E1462, 2014.

[50] P. Liu and L. Yin. Spontaneous facial expression analysis based on temperature changes and head motions. In *Proc. 11th IEEE Int. Conf. and Workshops Automatic Face and Gesture Recognition (FG)*, volume 1, pages 1–6, May 2015.

[51] Michael Lyons, Shigeru Akamatsu, Miyuki Kamachi, and Jiro Gyoba. Coding facial expressions with gabor wavelets. In *Proceedings Third IEEE international conference on automatic face and gesture recognition*, pages 200–205. IEEE, 1998.

[52] T. Kanade and J. F. Cohn. Comprehensive database for facial expression analysis. In *Proc. Fourth IEEE Int. Conf. Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pages 46–53, March 2000.

[53] Maja Pantic, Michel Valstar, Ron Rademaker, and Ludo Maat. Web-based database for facial expression analysis. In *2005 IEEE international conference on multimedia and Expo*, pages 5–pp. IEEE, 2005.

[54] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition - Workshops*, pages 94–101, June 2010.

[55] Niki Aifanti, Christos Papachristou, and Anastasios Delopoulos. The mug facial expression database. In *11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10*, pages 1–4. IEEE, 2010.

[56] Oliver Langner, Ron Dotsch, Gijsbert Bijlstra, Daniel H. J. Wigboldus, Skyler T. Hawk, and Ad van Knippenberg. Presentation and validation of the radboud faces database. *Cognition and Emotion*, 24:1377–1388, 2010.

[57] Deepali Aneja, Alex Colburn, Gary Faigin, Linda Shapiro, and Barbara Mones. Modeling stylized character expressions via deep learning. In *Asian Conference on Computer Vision*, pages 136–153. Springer, 2016.

[58] Steven R Livingstone and Frank A Russo. The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. *PloS one*, 13(5):e0196391, 2018.

[59] Paul Ekman, Wallace V. Friesen, and Joseph C. Hager. *Facial action coding system*. Research Nexus, Salt Lake City, Utah, 2002.

[60] P Ekman and W V Friesen. Constants across cultures in the face and emotion. *Journal of personality and social psychology*, 17:124–129, February 1971.

[61] P Ekman. Strong evidence for universals in facial expressions: a reply to russell's mistaken critique. *Psychological bulletin*, 115:268–287, March 1994.

[62] David Matsumoto. More evidence for the universality of a contempt expression. *Motivation and Emotion*, 16(4):363, December 1992.

[63] G. Li, X. Cai, X. Li, and Y. Liu. An efficient face normalization algorithm based on eyes detection. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 3843–3848, October 2006.

[64] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE Computer Society, 2005.

[65] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, September 2010.

[66] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1867–1874, June 2014.

[67] Christos Sagonas, Epameinondas Antonakos, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 faces in-the-wild challenge: Database and results. *Image and vision computing*, 47:3–18, 2016.

[68] Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv e-prints*, page arXiv:1212.5701, Dec 2012.

[69] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[70] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 254–263, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[71] Yan Yan, Rómer Rosales, Glenn Fung, Subramanian Ramanathan, and Jennifer G. Dy. Learning from multiple annotators with varying expertise. *Machine Learning*, 95(3):291–327, 2014.

[72] Julián Gil González, Andrés Marino Álvarez-Meza, and Alvaro Angel Orozco Gutierrez. Learning from multiple annotators using kernel alignment. *Pattern Recognition Letters*, 116:150–156, 2018.

[73] Chirine Wolley and Mohamed Quafafou. Learning from multiple naive annotators. In Shuigeng Zhou, Songmao Zhang, and George Karypis, editors, *Advanced Data Mining and Applications, 8th International Conference, ADMA 2012, Nanjing, China, December 15-18, 2012. Proceedings*, volume 7713 of *Lecture Notes in Computer Science*, pages 173–185. Springer, 2012.

[74] Yohko Maki, Hiroshi Yoshida, Tomoharu Yamaguchi, and Haruyasu Yamaguchi. Relative preservation of the recognition of positive facial expression "happiness" in alzheimer disease. *International psychogeriatrics*, 25:105–110, January 2013.

[75] Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bullettin*, 76(5):378–382, 1971.

[76] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, April 1960.

Appendices

Appendix A

Normalization algorithm and GUI

In order to determine the best possible parameters for the normalization algorithm described in Section 2.2.1, a graphical user interface was created: it can be seen in Figure A.1.

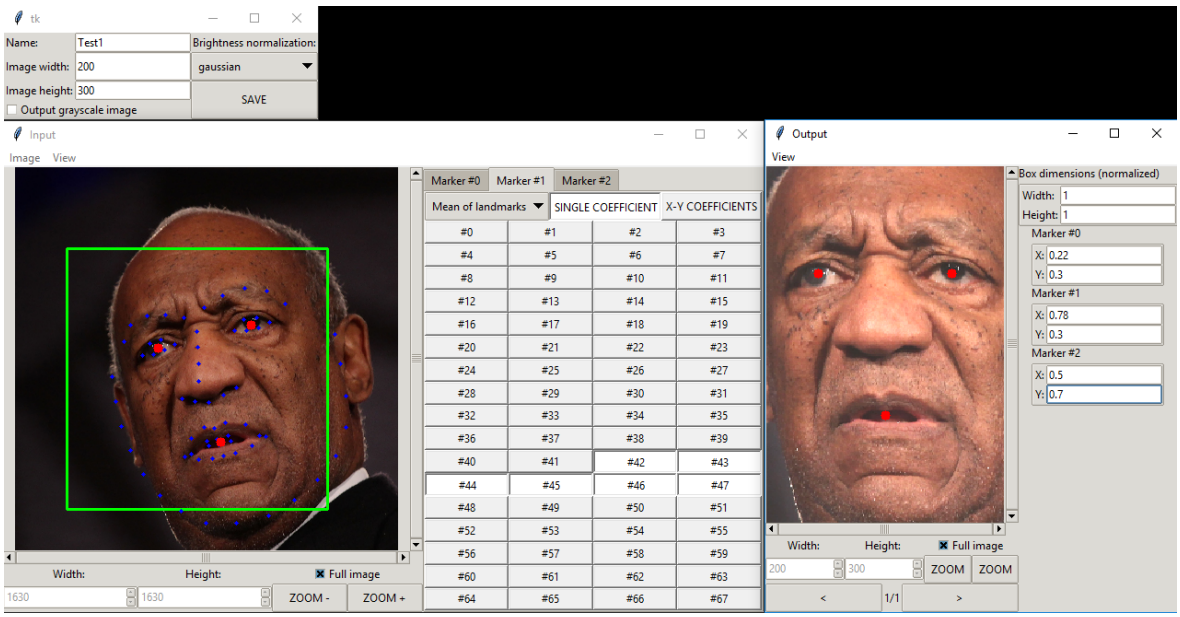


Figure A.1: *Graphical user interface used to choose the parameters of the normalization algorithm. The algorithm is applied on the fly (even with real-time input) while editing the parameters and it can then be saved to be used in a real application. The parameters shown are the ones that were actually used.*

As the screenshot shows, the interface is made up of three windows:

- a small main window, used for setting all parameters not related to the markers position
- an input window, used to select the position of the markers in the input (real)

image

- an output window, used to select the position of the markers in the output (normalized) image

A.1 Main window

The only really important parameter that can be set here is the brightness normalization algorithm to be used. The possibilities are:

- histogram equalization
- linear normalization
- Gaussian normalization

The *Gaussian* normalization algorithm is described in Section 2.2.1 and it is the one used in the final application.

In this window it is also possible to set the output shape of the image as width, height and color depth (either 8 bit grayscale or 24 bit BGR). This choice reflects in the output image shown in the interface and it is used as the default output shape, but it can be dynamically changed to fit the input shape of the used neural network without changing the algorithm. In fact, the various networks that were trained had different input shapes but the same algorithm was always used.

The *Name* option is used to associate a name to the algorithm, so that it can be registered as a default algorithm in the `cicc.normalization` module.

A.2 Input window

This window allows to find the best linear combination of landmarks for defining the markers position. From the *Input* menu it is possible to select an image to be shown, either an image file stored in the computer or a real-time video capture from a webcam (if the computer has one). The *View* menu lets the user choose to show the bounding box of all automatically detected faces, the position of the 68 landmarks and the 3 markers.

On the right of the input image there is a tab for each marker that lets the user define their position in three possible ways:

Single landmark: use the position of a landmark as a marker position too

Mean of landmarks: use the average position of various landmarks as a marker position

Full matrix: define all the coefficients of the linear combination of the various landmarks

It is also possible to decide whether to define the X and Y position of the marker based on different landmarks.

If l_x, l_y are 68×1 vectors containing, respectively, the x and y coordinates of the landmarks and X, Y are two 3×68 matrices containing the coefficients selected in the right panel of the input window relative to the X and Y markers position (they are equal if the "Single coefficient" option is selected), it is possible to define the markers position on the input image as:

$$m_x = Xl_x$$

$$m_y = Yl_y$$

A.3 Output window

This window shows the result of the normalization and allows for the user to define the position of the markers on the normalization template. As in the input window, the *View* menu is used to show or hide the markers on the image. If more than one face is detected in the image, all faces can be seen one at a time by pressing on the arrows below the output image.

Once the markers position on both the input image (see above) and on the template (let the coordinates, appropriately scaled to the chosen output shape, be denoted by t_x and t_y) have been determined, using OpenCV functions it is possible to find a 3×3 matrix A such that:

$$\left[\begin{array}{c|c} t_x & t_y \end{array}\right] = A \left[\begin{array}{c|c} m_x & m_y \end{array}\right]$$

This matrix can then be used to perform an affine transformation of the input image and obtain an aligned output image.

The chosen parameters can be found in Table A.1.

Table A.1: *Chosen parameters for the normalization algorithm*

Parameters from main window			
Image shape	Width: 200	Height: 300	Depth: Full color
Brightness normalization	Gaussian		

Parameters from input window	
Marker #0	Average of landmarks #36 through #41
Marker #1	Average of landmarks #42 through #47
Marker #2	Average of landmarks #52 through #67

Parameters from output window		
Template shape	Width: 1	Height: 1
Marker #0	X: 0.22	Y: 0.30
Marker #1	X: 0.78	Y: 0.30
Marker #2	X: 0.50	Y: 0.70

Appendix B

`netmodeler` interface

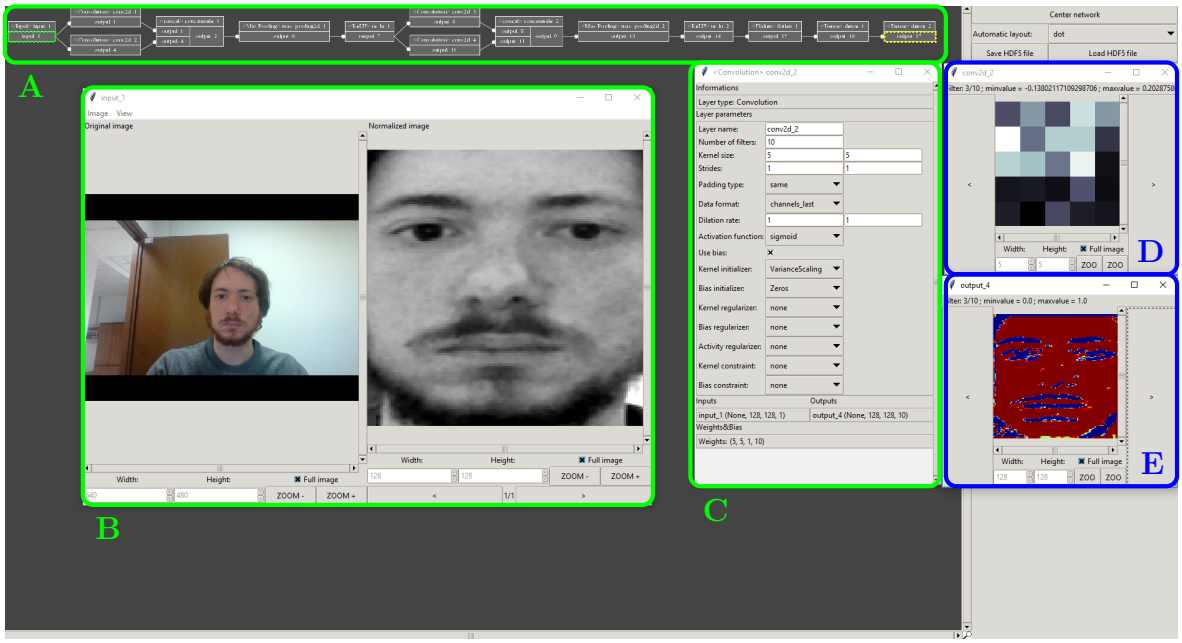


Figure B.1: *Screenshot from the* `netmodeler` *application. On the top of the main window (A), the network topology can be seen. The other windows show:*

B *input image, both as taken in real time from a webcam and normalized (the latter one is the one supplied to the network)*

C *information about a convolutional layer with its name, all its editable parameters, the incoming/outgoing tensors and its trained weights*

D *visualization in grayscale of the kernel weights of the 3rd filter in the layer*

E *visualization in false colors of the activation map of the 3rd filter in the layer*

Figure B.1 shows a screenshot of an application that was created to simplify the creation, visualization and debugging of neural networks. This application, called

`netmodeler`, was written in Python and offers a simple and intuitive GUI to many functionalities provided by Keras in the context of this project. With it, it is possible to visualize a neural network as a graph of connected nodes (positioned either manually or automatically through the use of the Graphviz[1] graph visualization software suite), with indications of the layer type, the connections between layers and the (possibly multiple) outputs of the layers that can also be marked as inputs (green) or outputs (yellow) of the network; Figure B.2 shows a detail of this.

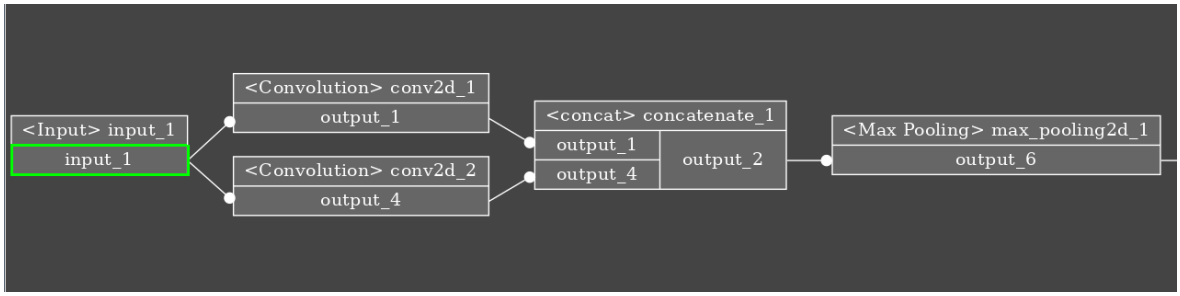


Figure B.2: *Detail of the network topology visualization in the `netmodeler` GUI.*

By double clicking on the layer names, it is possible to open a window containing all the parameters of that layer that can be easily edited from there; this window also shows all the input and output tensors of the layer with their respective shapes and, for *Convolution* and *Dense* layers, the shape of the weights tensor. By then clicking on the tensors names (either form the previous window or directly on the topology visualization) a different window is opened showing a graphical representation of their values in real time (if an appropriate input was selected from the window opened by clicking on the input tensors). The input image can be either an image stored on the PC or pictures captured in real time from a webcam.

At the current stage of development, the interface does not allow for the addition of layers to a network or its training, but it is possible to load a HDF5 file containing a neural network built and possibly trained with Keras, edit it with this GUI and then save it again.

[1] `http://www.graphviz.org`

Appendix C

Neural networks parameters

C.1 Architecture parameters

Here are reported the parameters used for building and training the various neural networks used in this project. The networks which name starts with "`test`" were trained using the AffectNet training set described in Section 2.1.1, while the ones which name starts with "`Compare`" were trained on the CK+ dataset described in Section 2.1.1.

The parameters that are common to all networks and special cases are described in Section 2.2.2 and Section 2.2.3.

Table C.1: *Basic structure information of the used network architectures. A depth of* 1 *indicates a grayscale input, while it is* 3 *for BGR color images.*

	Input shape				**Total number**
Network	*Width*	*Height*	*Depth*	**CMR layers**	**of parameters**
`test0`	200	300	3	1	19670488
`test1`	200	300	1	1	19670184
`test2`	200	200	3	1	13116888
`test3`	200	200	1	1	13116584
`test4`	200	300	1	1	19670184
`test5`	200	200	1	1	13116584
`test6`	128	128	1	1	5129384
`test7`	64	64	1	1	1189032
`test8`	200	300	1	2	797040
`test9`	200	200	1	2	534896
`test10`	128	128	1	2	215408
`test11`	64	64	1	2	43376

(continues in next page)

Table C.1: *Basic structure information of the used network architectures. A depth of* 1 *indicates a grayscale input, while it is* 3 *for BGR color images.*

(continued from last page)

	Input shape				Total number
Network	*Width*	*Height*	*Depth*	**CMR layers**	**of parameters**
test12	200	300	1	1	797040
test13	128	128	1	1	5129384
test14	128	128	1	1	5129384
test15	128	128	1	0	5129384
test16	128	128	1	1	10249544
test17	128	128	1	1	33563816
test18	128	128	1	1	10258600
test19	128	128	1	2	16789048
test20	128	128	1	2	16791640
test21	128	128	1	2	33568712
test22	128	128	1	2	15756856
test23	128	128	1	2	8395832
test24	256	256	1	3	16801240
test25	256	256	1	3	16801240
test26	256	256	1	2	12856888
test27	256	256	1	2	12856888
Compare1	64	64	1	1	56486
Compare2	128	128	1	1	241190
Compare3	128	128	1	2	61742
Compare4	128	128	1	1	7334

Table C.2: *Parameters used in the first CMR layer of each network architecture*

	Filters		Max Pooling	
Network	3×3	5×5	*Pool size*	*Pool stride*
test0	3	5	5	5
test1	3	5	5	5
test2	3	5	5	5
test3	3	5	5	5
test4	3	5	5	5
test5	3	5	5	5
test6	3	5	5	5

(continues in next page)

Table C.2: *Parameters used in the first CMR layer of each network architecture*

(continued from last page)

	Filters		**Max Pooling**	
Network	*3 × 3*	*5 × 5*	*Pool size*	*Pool stride*
test7	3	5	5	5
test8	3	5	5	5
test9	3	5	5	5
test10	3	5	5	5
test11	3	5	5	5
test12	3	5	5	5
test13	3	5	5	5
test14	3	5	5	5
test16	6	10	5	5
test17	3	5	2	2
test18	3	5	5	5
test19	3	5	2	2
test20	6	10	2	2
test21	3	5	2	2
test22	3	5	3	2
test23	3	5	2	2
test24	3	5	2	2
test25	3	5	2	2
test26	3	5	3	3
test27	3	5	3	3
Compare1	3	0	5	5
Compare2	3	0	5	5
Compare3	3	0	2	2
Compare4	3	0	10	10

Table C.3: *Parameters used in the second CMR layer of each network architecture*

	Filters		**Max Pooling**	
Network	*3 × 3*	*5 × 5*	*Pool size*	*Pool stride*
test8	3	5	5	5
test9	3	5	5	5
test10	3	5	5	5
test11	3	5	5	5

(continues in next page)

Table C.3: *Parameters used in the second CMR layer of each network architecture*

(continued from last page)

	Filters		Max Pooling	
Network	3×3	5×5	*Pool size*	*Pool stride*
test19	6	10	2	2
test20	6	10	2	2
test21	12	20	2	2
test22	6	10	3	2
test23	6	10	2	2
test24	12	20	2	2
test25	12	20	2	2
test26	6	10	3	3
test27	6	10	3	3
Compare3	3	0	2	2

Table C.4: *Parameters used in the third CMR layer of each network architecture*

	Filters		Max Pooling	
Network	3×3	5×5	*Pool size*	*Pool stride*
test24	6	10	2	2
test25	6	10	2	2

Table C.5: *Parameters of the fully connected layer of each network architecture.*

Network	Number of neurons	Activation function
test0	1024	relu
test1	1024	relu
test2	1024	relu
test3	1024	relu
test4	1024	sigmoid
test5	1024	sigmoid
test6	1024	relu
test7	1024	relu
test8	1024	relu
test9	1024	relu
test10	1024	relu
test11	1024	relu

(continues in next page)

Table C.5: *Parameters of the fully connected layer of each network architecture.*

(continued from last page)

Network	Number of neurons	Activation function
test12	1024	relu
test13	1024	relu
test14	1024	relu
test15	-	-
test16	1024	relu
test17	1024	relu
test18	2048	relu
test19	1024	relu
test20	1024	relu
test21	1024	relu
test22	1024	relu
test23	512	relu
test24	1024	relu
test25	1024	relu
test26	1024	relu
test27	1024	relu
Compare1	128	relu
Compare2	128	relu
Compare3	20	relu
Compare4	128	relu

C.2 Training parameters

Table C.6 shows the parameters used to train all the networks. The first column indicates the network architecture used, as defined by Section 2.2.2 and Section C.1; the second one shows how many times the network was trained with the following parameters; the last two columns show the batch size and number of epochs of training (remembering that 10000 samples were used from each epoch, as indicated in Section 2.2.2).

When multiple values for the last two columns are provided in the same row, it means that the training was done in sequential phases, changing the parameters between them.

Table C.6: *Training parameters used for the various networks*

Network	Repetitions	Batch size	Epochs
test0	1	512	100
test0	1	1000	100
test0	1	64	100
test1	1	512	100
test1	1	1000	100
test1	1	64	100
test2	1	512	100
test2	1	1000	100
test2	1	64	100
test3	1	512	100
test3	1	1000	100
test3	1	64	100
test4	1	512	100
test4	1	1000	100
test4	1	64	100
test5	1	512	100
test5	1	1000	100
test5	1	64	100
test6	1	512	100
test6	1	1000	100
test6	1	64	100
test7	1	512	100
test7	1	1000	100
test7	1	64	100
test8	3	64	100
test9	3	64	100
test10	1	512	100
test10	1	1000	100
test10	1	64	100
test11	1	512	100
test11	1	1000	100
test11	1	64	100
test12	1	512	100

(continues in next page)

Table C.6: *Training parameters used for the various networks*

(continued from last page)

Network	Repetitions	Batch size	Epochs
test12	1	1000	100
test12	1	64	100
test13	3	64	100
test14	3	1000	50
test15	3	64	50
test16	3	1000	50
		64	50
test17	3	1000	50
		64	50
test18	3	1000	50
		64	50
test19	3	1000	50
		64	50
test20	3	1000	50
		64	50
test21	3	1000	50
		64	50
test22	3	1000	50
		64	50
test23	3	1000	50
		64	50
test24	3	32	100
test25	3	32	100
test26	3	32	100
test27	3	32	100
Compare1	3	64	100
Compare2	3	64	100
Compare3	3	64	100
Compare4	3	64	100

Appendix D

Mathematical definitions

D.1 Fuzzy Categorical Cross Entropy

The below described loss function, called Fuzzy Categorical Cross Entropy for reasons that will become clear later and denoted by the symbol $\mathcal{F}$, uses a geometrical approach to overcome the limitations of the MSE loss function and exploit the properties and peculiarities of a classification problem where both the ground truth and the predictions are given as estimations of the membership functions of a fuzzy partition. The classes are considered as axes in an euclidean space. This way, the ground truth and prediction vectors, defined as

$$\mathbf{T} = [t_1, \ldots, t_C] = [\mu_1(i), \ldots, \mu_C(i)]^T$$

$$\mathbf{P} = [p_1, \ldots, p_C] = [f_1(i), \ldots, f_C(i)]^T$$

where $\mu_c(i)$ and $f_c(i)$ are respectively the ground truth value and the prediction for a certain class relative to an image, are vectors in this space with the following properties (here $\mathbf{V}$ denotes both $\mathbf{T}$ and $\mathbf{P}$):

$$\|\mathbf{V}\|_1 = \sum_{c=1}^{C} v_c = 1 \qquad \text{(i.e. the vectors sit on the surface of a } C\text{-orthoplex)}$$

$$\forall c = 1, \ldots, C \quad v_c \geq 0 \quad \text{(i.e. the vectors are contained in the nonnegative orthant)}$$

The error of a prediction can then be correlated to the angle between these two vectors: in fact, when the ground truth and the prediction are perfectly equal the angle between $\mathbf{T}$ and $\mathbf{P}$ is zero, while when none of the predicted labels correspond to a correct label the two vectors are orthogonal.

Instead of using the original vectors, however, these slightly different versions are

defined:

$$\hat{\mathbf{T}} = [\hat{t}_1, \ldots, \hat{t}_C] = \left[\sqrt{\mu_1(i)}, \ldots, \sqrt{\mu_C(i)}\right]^T$$

$$\hat{\mathbf{P}} = [\hat{p}_1, \ldots, \hat{p}_C] = \left[\sqrt{f_1(i)}, \ldots, \sqrt{f_C(i)}\right]^T$$

with the following properties:

$$\left\|\hat{\mathbf{V}}\right\|_2 = \sum_{c=1}^{C} \hat{v}_c^2 = 1 \qquad \text{(i.e. the vectors sit on the surface of a } (C-1)\text{-sphere)}$$

$$\forall c = 1, \ldots, C \quad \hat{v}_c \geq 0 \quad \text{(i.e. the vectors are contained in the nonnegative orthant)}$$

Applying this transformation changes the angles, as shown in Figure D.1, but they are still suitable to evaluate a prediction.

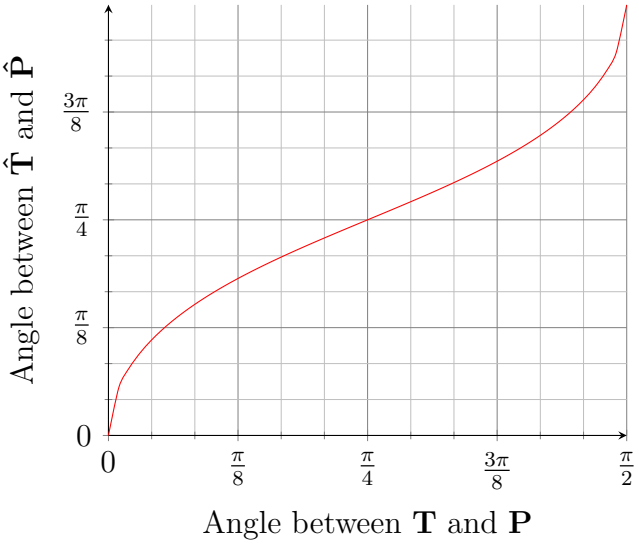


Figure D.1: *Angles modification when the square root of the components is taken*

The easiest measure related to the angle between the two vectors that can be computed is the cosine of the angle:

$$\hat{\mathbf{T}} \cdot \hat{\mathbf{P}} = \sum_{c=1}^{C} \hat{t}_c \hat{p}_c = \overset{1}{\cancel{\left\|\hat{\mathbf{T}}\right\|_2}} \overset{1}{\cancel{\left\|\hat{\mathbf{P}}\right\|_2}} \cos\left(\angle\left(\hat{\mathbf{T}}, \hat{\mathbf{P}}\right)\right) = \cos\left(\angle\left(\hat{\mathbf{T}}, \hat{\mathbf{P}}\right)\right)$$

According to the above relation, the scalar product of the two vectors can be used, giving a result of 0 for orthogonal vectors and 1 for completely identical vectors. The error can then be computed by a suitable loss function such that:

$$\text{loss}|_{\hat{\mathbf{T}} \cdot \hat{\mathbf{P}} = 1} = 0$$

$$a < b \Rightarrow \text{loss}|_{\hat{\mathbf{T}}\cdot\hat{\mathbf{P}}=a} > \text{loss}|_{\hat{\mathbf{T}}\cdot\hat{\mathbf{P}}=b}$$

This also shows why the square root of the components was taken: if the original truth/prediction values were used, the result would be influenced by the dispersion of the values in the vectors. Crisp one-hot vectors would have a modulus of 1 while more uncertain classifications would decrease the value of the modulus and hence of the scalar product which in turn means a higher loss, so that when an image has an uncertain ground truth even a perfect prediction would give a non-zero loss. This problem could be overcome by simply dividing the scalar product by the product of the two norms, but considering that these operations are supposed to be executed on a GPU the proposed method is more computationally efficient since all components can be considered independent (and thus the operations can be parallelized) except for the last sum that should still be performed. In addition, using these vectors the loss can be defined as:

$$\mathcal{F}(\mathbf{T},\mathbf{P}) = -\log\left(\hat{\mathbf{T}}\cdot\hat{\mathbf{P}}\right)^2 = -2\log\left(\sum_{c=1}^{C}\sqrt{t_c}\sqrt{p_c}\right) = -2\log\left(\sum_{c=1}^{C}\sqrt{t_c p_c}\right)$$

which can be shown to be equivalent to categorical cross entropy when one-hot ground truths are provided. In fact, considering without loss of generality that the first class is the correct one:

$$\mathcal{F} = -2\log\left(\sum_{c=1}^{C}\sqrt{t_c p_c}\right) = -\log\left(\sqrt{1\cdot p_1} + \cancel{\sum_{c=2}^{C}\sqrt{0\cdot p_c}}\right)^2 = -\log p_1$$

$$CE = -\sum_{c=1}^{C} t_c \log p_c = -\left(1\cdot\log p_1 + \cancel{\sum_{c=2}^{C} 0\cdot\log p_c}\right) = -\log p_1$$

This loss function evaluates to 0 in the case of a perfect prediction and $+\infty$ when the image is predicted to belong only to classes that have a true label of 0. Similarly to the classical categorical cross entropy care should be taken when using this loss function to avoid that the argument of the logarithm is ever equal to zero, either through the use of some sort of regularization of the prediction (e.g. applying the softmax function to a prediction vector ensures that all components of the vector are strictly greater than zero) or by modifying the loss function as:

$$\mathcal{F} = -2\log\left(\sum_{c=1}^{C}\sqrt{t_c p_c} + \epsilon\right) \qquad \text{(with } \epsilon > 0\text{)}$$

so that:

$$\mathcal{F}|_{\hat{\mathbf{T}}\cdot\hat{\mathbf{P}}=1} = -2\log(1+\epsilon) \approx -2\epsilon \approx 0 \qquad \text{(for small } \epsilon\text{)}$$

$$\mathcal{F}|_{\hat{\mathbf{T}}\cdot\hat{\mathbf{P}}=0} = -2\log(\epsilon) < +\infty$$

D.1.1 Single label prediction

Sometimes it is required to extract a single representative class from the prediction vector. In such cases, there is not much interest in approximating the membership function for all classes as long as a "probable" single class is selected. It is thus better to have a *crisp* output that does not correctly estimate the various values of the target while still selecting a class with a high membership function than to get near the target while having a maximum value corresponding to a class that has a very low membership function.

For this kind of application the $\mathcal{F}$ loss function is thus modified, calling the new version $\mathcal{F}^*$, in order to have the following properties:

1. $\mathcal{F}^* = 0$ when the prediction vector equals 1 for the class with maximum membership function and 0 otherwise
2. $\mathcal{F}^*$ is lower for prediction vectors with crisp outputs

When the hypothesis of property 1 is verified:

$$\mathcal{F} = -2\log\left(\sqrt{1\cdot\max_c t_c}\right) = -\log\left(\max_c t_c\right) =$$

$$= \begin{cases} -2\log\frac{1}{C} = 2\log C & \text{maximum uncertainty} \\ -2\log 1 = 0 & \text{minimum uncertainty} \end{cases}$$

So the property can be obtained by simply removing a per-target offset.

Then, an additional component can be added to the loss function in order to increase the loss for predictions that are more "uncertain". As Section D.3.3 shows, it is possible to use the squared norm of the vector to have a measure of certainty and thus one minus it as a measure of uncertainty.

The resulting loss function is:

$$\begin{aligned}\mathcal{F}^* &= -2\log\left(\sum_{c=1}^{C}\sqrt{t_c p_c}\right) - \left(-\log\left(\max_c t_c\right)\right) + \alpha\left(1-\sum_{c=1}^{C}p_c^2\right) = \\ &= -2\log\left(\frac{\sum_{c=1}^{C}\sqrt{t_c p_c}}{\sqrt{\max_c t_c}}\right) + \alpha\left(1-\sum_{c=1}^{C}p_c^2\right) = \\ &= -2\log\left(\sum_{c=1}^{C}\sqrt{\frac{t_c}{\max_c t_c}p_c}\right) + \alpha\left(1-\sum_{c=1}^{C}p_c^2\right)\end{aligned}$$

where α is a parameter that indicates how important is it that predictions are crisp. Since the ground truths are known a priori by definition, it is possible to define a corrected ground truth vector $\mathbf{T}^*$, where $t_c^* = \frac{t_c}{\max_c t_c}$, which is computed once for each target at the start of training. The loss function is then:

$$\mathcal{F}^* = -2\log\left(\sum_{c=1}^{C}\sqrt{t_c^* p_c}\right) + \alpha\left(1-\sum_{c=1}^{C}p_c^2\right)$$

D.2 Confusion/Coincidence matrices

A *confusion matrix* is a very common tool used to analyse the performance in binary classification tasks, comparing the output of a predictor to the ground truth. Here, a generalization is described in order to expand the concept to multi-class classification and with multiple predictors/annotators.

D.2.1 Multiple classes

Let N be the number of items on which the classification is performed C the number of classes and $t_i, p_i \in \{1,\dots,C\}$ the class to which item i is assigned, respectively, by the ground truth and by the predictor. Let then:

$$t_{i=c} = \begin{cases}1 & \text{if } t_i = c \\ 0 & \text{otherwise}\end{cases} \quad\Bigg|\quad p_{i=c} = \begin{cases}1 & \text{if } p_i = c \\ 0 & \text{otherwise}\end{cases}$$

The confusion matrix M is then a $C \times C$ matrix where each element is defined as:

$$m_{c_t c_p} = \sum_{i=1}^{N} t_{i=c_t} p_{i=c_p}$$

Additionally, the following are defined:

$$m_{c_t\cdot} = \sum_{c_p=1}^{C} m_{c_t c_p} \;\Big|\; m_{\cdot c_p} = \sum_{c_t=1}^{C} m_{c_t c_p}$$

When $C = 2$, the above definition is equal to the standard one.

From this matrix, various metrics can be computed:

$$\text{precision}(c) = \frac{\text{items correctly classified in class } c}{\text{all items classified in class } c} = \frac{m_{cc}}{m_{\cdot c}}$$

$$\text{recall}(c) = \text{sensitivity}(c) = \frac{\text{items correctly classified in class } c}{\text{all items of class } c} = \frac{m_{cc}}{m_{c\cdot}}$$

$$\text{specificity}(c) = \frac{\text{items correctly classified outside class } c}{\text{all items classified outside class } c} = \frac{\sum_{k\neq c} m_{\cdot k} - m_{ck}}{\sum_{k\neq c} m_{\cdot k}}$$

$$\mathrm{F}_\beta(c) = \left(1+\beta^2\right) \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} = \left(1+\beta^2\right) \frac{m_{cc}}{\beta^2 m_{c\cdot} + m_{\cdot c}}$$

(most commonly the F_1 score is used)

$$\begin{aligned}\text{accuracy}(c) &= \frac{\text{items correctly classified in class } c + \text{items correctly classified outside class } c}{\text{all items}} \\ &= \frac{m_{cc} + \sum_{k\neq c} m_{\cdot k} - m_{ck}}{N}\end{aligned}$$

For all of this metrics, a line above the name indicates the mean among all classes, except for the accuracy where the global version is defined as:

$$\overline{\text{accuracy}} = \frac{\text{correctly classified items}}{\text{all items}} = \frac{\sum_{c=1}^{C} m_{cc}}{N}$$

D.2.2 Multiple predictors

In order to extend the above definitions to the case in which multiple predictors are compared simultaneously to the same ground truth, all the various predictors can be considered as a single entity and the annotations to a same item can be treated as if they were done to different items. The ground truth annotations are then repeated in order to include all these virtual items. At this point, the confusion matrix can be computed normally.

Let N be the number of items on which the classification is performed C the number of classes, P the number of predictors and $t_i, p_{ij} \in \{1, \ldots, C\}$ the class to which item i is assigned, respectively, by the ground truth and by the jth predictor. Also define $t_{i=c}$ and $p_{ij=c}$ similarly to before.

The confusion matrix M is then a $C \times C$ matrix where each element is defined as:

$$m_{c_t c_p} = \sum_{j=1}^{P} \sum_{i=1}^{N} t_{i=c_t} p_{ij=c_p} = \sum_{i=1}^{N} t_{i=c_t} \sum_{j=1}^{P} p_{ij=c_p}$$

All the previously defined metrics are applicable here too.

D.2.3 Multiple annotators

When multiple annotators are considered the distinction between ground truth and predictor is lost. The *coincidence matrix* is hence introduced, which is because of this symmetrical.

Let N be the number of items on which the classification is performed, C the number of classes and n_{ic} the number of annotators that assigned item i to class c. Let then $n_{i\cdot} = \sum_{c=1}^{C} n_{ic}$ for $i = 1, \ldots, N$ be the number of annotations available for item i and $n_{\cdot c} = \sum_{i=1}^{N} n_{ic}$ for $c = 1, \ldots, C$ the total number of annotations assigned to class c.

The coincidence matrix M is then a $C \times C$ matrix where each element is defined as:

$$m_{c_1 c_2} = \sum_{i=1}^{N} \cancel{n_{i\cdot}} \frac{\pi_{ic_1c_2}}{\cancel{n_{i\cdot}}\,(n_{i\cdot} - 1)} = \sum_{i=1}^{N} \frac{\pi_{ic_1c_2}}{n_{i\cdot} - 1}$$

where $\pi_{ic_1c_2}$ is the number of ordered pairs of annotations of item i for which one annotations assigns the item to class c_1 and the other assigns it to class c_2. It is defined as:

$$\pi_{ic_1c_2} = \begin{cases} n_{ic_1}\,(n_{ic_1} - 1) & \text{if } c_1 = c_2 \\ n_{ic_1} n_{ic_2} & \text{if } c_1 \neq c_2 \end{cases}$$

Note that the fact that the pairs are *ordered* is not important since the order disappears when dividing this number by the number of all ordered pairs of annotations of item i, i.e. $n_{i\cdot}\,(n_{i\cdot} - 1)$, the order disappears.

The coincidence matrix thus shows the proportion with which two annotations for the same items assign it to two specific (different or equal) classes, rescaled so that:

$$m_{c_1\cdot} = m_{\cdot c_1} = \sum_{c_2=1}^{C} m_{c_1c_2} = n_{\cdot c_1}$$

and

$$\sum_{\substack{c_1=1 \\ c_2=1}}^{C} m_{c_1c_2} = \sum_{c=1}^{C} n_{\cdot c} = \sum_{i=1}^{N} n_{i\cdot} = \text{total number of annotations}$$

An *expected coincidence matrix* can also be defined, by assuming that all items present the same distribution of labels, i.e. by considering all annotations as if they were relative to a single item. Let this matrix be called E. By replacing $\pi_{ic_1c_2}$ with:

$$\hat{\pi}_{c_1c_2} = \begin{cases} n_{\cdot c_1}\left(n_{\cdot c_1} - 1\right) & \text{if } c_1 = c_2 \\ n_{\cdot c_1} n_{\cdot c_2} & \text{if } c_1 \neq c_2 \end{cases}$$

and $n_{i\cdot}$ with $n_{\cdot\cdot} = \sum_{i=1}^{N} c_{i\cdot}$, the elements of the matrix can be defined as:

$$e_{c_1c_2} = \frac{\hat{\pi}_{c_1c_2}}{n_{\cdot\cdot} - 1}$$

The equivalent of the accuracy in confusion matrices can be called agreement in coincidence matrices, for reasons explained in Section D.3, and defined as:

$$\text{agreement} = \frac{\sum_{c=1}^{C} m_{cc}}{\sum_{i=1}^{N} n_{i\cdot}} = \frac{\sum_{c=1}^{C}\sum_{i=1}^{N} \frac{1}{n_{i\cdot}-1} n_{ic}\left(n_{ic}-1\right)}{\sum_{i=1}^{N} n_{i\cdot}} = \frac{\sum_{i=1}^{N} \frac{1}{n_{i\cdot}-1}\sum_{c=1}^{C} n_{ic}\left(n_{ic}-1\right)}{\sum_{i=1}^{N} n_{i\cdot}}$$

and similarly:

$$\text{expected agreement} = \frac{\sum_{c=1}^{C} e_{cc}}{n_{\cdot\cdot}} = \frac{\sum_{c=1}^{C} n_{\cdot c}\left(n_{\cdot c}-1\right)}{n_{\cdot\cdot}\left(n_{\cdot\cdot}-1\right)}$$

D.3 Annotators agreement metrics

D.3.1 From coincidence matrices

See Section D.2.3.

D.3.2 Fleiss' kappa with variable number of annotators

Fleiss' kappa, described in [75] and based on Cohen's kappa [76], is the most common statistic used to evaluate inter-rater agreement in tasks using nominal scales. A limitation of Fleiss' kappa is that it requires a fixed number of annotators for each annotated item. Here a generalization is presented while keeping the same meaning and values in the case of constant number of annotators, following the description given in the original paper.

Let N represent the total number of items in the dataset and k the number of classes into which assignments are made. Define n_i to be the number of annotators who annotated the item i and n_{ij} to be how many of those annotators assigned the ith item to the jth class. Then the proportion of all assignments which were to the jth

class is:

$$p_j = \frac{\sum_{i=1}^{N} n_{ij}}{\sum_{i=1}^{N} n_i}$$

with $\sum_{j=1}^{k} p_j = 1$.

The extent of agreement among the n_i annotators for the ith item may be measured by the proportion of agreeing pairs out of all the $n_i(n_i-1)$ possible pairs of annotations:

$$P_i = \frac{1}{n_i(n_i-1)} \sum_{j=1}^{k} n_{ij}(n_{ij}-1)$$

The overall extent of agreement may then be measured by the mean of the P_is, weighted by the number of annotations for each image:

$$\bar{P} = \frac{\sum_{i=1}^{N} P_i n_i}{\sum_{i=1}^{N} n_i} = \frac{\sum_{i=1}^{N} \frac{1}{\cancel{n_i}(n_i-1)} \sum_{j=1}^{k} n_{ij}(n_{ij}-1)\cancel{n_i}}{\sum_{i=1}^{N} n_i} = \frac{\sum_{i=1}^{N} \frac{1}{n_i-1} \sum_{j=1}^{k} n_{ij}(n_{ij}-1)}{\sum_{i=1}^{N} n_i}$$

This represents the probability that, taking an random item and two random annotators who annotated that item, the two annotators agree. It can be compared with the accuracy of a classifier and it is equivalent to the *agreement* metric defined in Section D.2.3 from the coincidence matrix.

Some degree of agreement, however, is to be expected solely on the basis of chance. In fact, if the annotators made their assignments purely at random, one would expect the mean proportion of agreement to be:

$$\bar{P}_e = \sum_{j=1}^{k} p_j^2$$

This is very similar to the expected agreement described in Section D.2.3, except that here the number of annotations is not taken into account. The two formulations are equivalent when the number of annotations approaches infinity.

The quantity $1 - \bar{P}_e$ measures the degree of agreement attainable over and above what would be predicted by chance, while the degree of agreement actually attained in excess of chance is $\bar{P} - \bar{P}_e$ so that a normalized measure of overall agreement, corrected for the amount expected by chance, is:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}$$

D.3.3 With fuzzy predictions

The degree of agreement between annotations can also be used as a measure of how certain or, conversely, ambiguous the classification of an item or a dataset is. It would thus be useful to extend this concept to predictions that define different membership values for each class in order to asses how confident the predictor is of its classification.

In order to mimic the definition of Fleiss' kappa described in Section D.3.2, let μ_{ij} be the value of the prediction on item i for class j and consequently $n_{ij} = \mu_{ij} n_i$. Since there are not real distinct annotations, let $n_i = n \ \forall i$ where n is an arbitrary large number.

Then:

$$p_j = \frac{\sum_{i=1}^{N} \mu_{ij} n}{\sum_{i=1}^{N} n} = \frac{\cancel{n} \sum_{i=1}^{N} \mu_{ij}}{\cancel{n} N} = \frac{1}{N} \sum_{i=1}^{N} \mu_{ij}$$

$$P_i = \frac{1}{\cancel{n}(n-1)} \sum_{j=1}^{k} \mu_{ij} \cancel{n} \mu_{ij} (\mu_{ij} n - 1) \underset{n \text{ large}}{\approx} \frac{1}{\cancel{n}} \sum_{j=1}^{k} \mu_{ij} \mu_{ij} \mu_{ij} \cancel{n} = \sum_{j=1}^{k} \mu_{ij}^2$$

This means that the squared norm of the prediction vector can be used as a measure of its confidence.

Continuing with the definition of Fleiss' kappa:

$$\bar{P} = \frac{\sum_{i=1}^{N} P_i n}{\sum_{i=1}^{N} n} = \frac{\cancel{n} \sum_{i=1}^{N} \sum_{j=1}^{k} \mu_{ij}^2}{\cancel{n} N} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{k} \mu_{ij}^2$$

$$\bar{P}_e = \sum_{j=1}^{k} p_j^2 = \sum_{j=1}^{k} \left(\frac{1}{N} \sum_{i=1}^{N} \mu_{ij} \right)^2 = \frac{1}{N^2} \sum_{j=1}^{k} \left(\sum_{i=1}^{N} \mu_{ij} \right)^2$$

These definitions can then be used to compute Fleiss' kappa normally since no reference to the fictitious n remain.